NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

EXPLORATORY ANALYSIS OF SUBMARINE TACTICS FOR MINE DETECTION AND AVOIDANCE

by

Terrence M. Nawara

September 2003

Thesis Advisor:	Steven E. Pilnick
Second Reader:	Arnold H. Buss

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Title (Mix case letters) Exploratory Analysis of Submarine Tactics for Mine Detection and Avoidance	**5. FUNDING NUMBERS**
6. AUTHOR(S) Terrence M. Nawara	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT `Approved for public release; distribution is unlimited`	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

This thesis provides an initial analytical basis for Tactical Decision Aids in submarine mine detection and avoidance (MDA). Five aspects of submarine MDA are studied. First, a network optimization model plans the best route through a minefield based on prior surveys of bottom clutter (NOn-mine Mine-like Bottom Objects, or NOMBOs). If a submarine is trying to avoid going through a minefield, the second model helps the submarine decide how far to back up if it detects a mine. A third model calculates minimum safe standoff distance for initiating submarine maneuvers around a given mine. This model takes into account submarine maneuvering characteristics and sensor error in the case of onboard sensor detection, or both navigation and mine location errors in the case of reported mine positions. The fourth aspect of the submarine MDA problem uses simulation to study the probability of safe transit based on alternative MDA tactics, various mine and NOMBO densities, and various probabilities of detection. Finally, the simulation examines the probability that a given MDA tactic will result in gridlock, i.e., the probability that a single attempt to penetrate the minefield is blocked by mines or NOMBOs.

14. SUBJECT TERMS Mine, Minefield, NOMBO, Mine Warfare, Mine Detection and Avoidance, Simulation, Poisson, Reactive, Path Planning, Obstacle Avoidance, Navigation, Mission Planning, Percolation	**15. NUMBER OF PAGES** 136
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

EXPLORATORY ANALYSIS OF SUBMARINE TACTICS FOR MINE DETECTION AND AVOIDANCE

Terrence M. Nawara
Lieutenant, United States Navy
B.S., United States Naval Academy, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2003

Author:	Terrence M. Nawara
Approved by:	Steven E. Pilnick Thesis Advisor
	Arnold H. Buss Second Reader
	James N. Eagle Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis provides an initial analytical basis for Tactical Decision Aids in submarine mine detection and avoidance (MDA). Five aspects of submarine MDA are studied. First, a network optimization model plans the best route through a minefield based on prior surveys of bottom clutter (NOn-mine Mine-like Bottom Objects, or NOMBOs). If a submarine is trying to avoid going through a minefield, the second model helps the submarine decide how far to back up if it detects a mine. A third model calculates minimum safe standoff distance for initiating submarine maneuvers around a given mine. This model takes into account submarine maneuvering characteristics and sensor error in the case of onboard sensor detection, or both navigation and mine location errors in the case of reported mine positions. The fourth aspect of the submarine MDA problem uses simulation to study the probability of safe transit based on alternative MDA tactics, various mine and NOMBO densities, and various probabilities of detection. Finally, the simulation examines the probability that a given MDA tactic will result in gridlock, i.e., the probability that a single attempt to penetrate the minefield is blocked by mines or NOMBOs.

THIS PAGE INTENTIONALLY LEFT BLANK

THESIS DISCLAIMER

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
- A. BACKGROUND 1
  - 1. Naval Mine Warfare History 1
  - 2. Availability of Mines 1
  - 3. Mine Detection and Avoidance 2
  - 4. Previous Studies 4
- B. OBJECTIVES 6

II. BATTLESPACE PREPARATION 7
- A. PROBLEM DEFINITION 7
- B. ANALYTICAL MODEL 8
- C. MODEL FORMULATION 9
  - 1. Non Linear Program 9
  - 2. Linear Program 10
- D. DETERMING PROBABILITY OF SAFE TRANSIT 11
- E. EXAMPLE MODEL IMPLEMENTATIONS 12
  - 1. Excel Implementation 12
  - 2. Excel Limitations 15
  - 3. GAMS Implementation 16
- G. BENEFITS OF MODEL 17

III. DISTANCE INTO MINEFIELD 19
- A. PROBLEM DEFINITION 19
- B. ASSUMPTIONS 20
- C. ANALYTICAL MODEL 20
  - 1. Variables 20
  - 2. Exponential Distribution 20
- D. TACTICAL DECISION AID 22
- E. CAUTION 23

IV. MINIMUM STANDOFF DISTANCE 25
- A. PROBLEM DEFINITION 25
- B. MODEL DESCRIPTION 25
  - 1. Calculating Lethal Range 25
  - 2. Avoiding a Mine Detected by Onboard Sonar 26
  - 3. Avoiding a Previously Reported Mine Position 28
- C. ANALYSIS 29
- D. RESULTS 30
  - 1. Minimum Standoff Distance When Detecting With Onboard Sonar 30
  - 2. Minimum Standoff Distance When Avoiding a Previously Reported Mine Position 31
  - 3. Impact on Sonar Range Performance 31

V. PROBABILITY OF SAFE TRANSIT 33
A. PROBLEM DEFINITION 33
B. MODELING ASSUMPTIONS 33
1. Environment 33
2. Mines 34
3. Submarine 34
C. MODEL DEVELOPMENT 36
1. Discrete-Event Simulation 36
2. Event Graphs and Simkit 36
3. Common Random Number Streams 38
D. SIMPLE MINEFIELD TRANSIT MODEL 38
1. Comparison to Analytical Model 38
2. Results of Comparison to SMT MODEL 41
E. LATERAL EXCURSION AVOIDANCE MANEUVER 42
1. Developing Lateral Excursion Avoidance Maneuver Model 42
2. Comparing LEAM to Other Models 46
3. New Scenarios 48
4. Exploration of LEAM Model 50

VI. GRIDLOCK 53
A. PROBLEM DEFINITION 53
B. MODEL DESCRIPTION 54
C. ANALYSIS 55
D. RESULTS 55

VII. CONCLUSIONS AND FUTURE WORK 59
A. CONCLUSIONS 59
B. RECOMMENDED FOLLOW ON RESEARCH 60
1. Unmanned Undersea Vehicle 60
2. 3-D Environment 60
3. Concurrently Avoiding ASW Contacts 60
4. Graphical User Interface and Clustering 61
5. Multiple Injuries Leading to a Kill 61
6. Exploring Battlespace Preparation 61
7. Percolation Theory 62

APPENDIX A. DEFINITION OF TERMS 63

APPENDIX B. SPATIAL POISSON PROCESS 65

APPENDIX C: GAMS BATTLESPACE PREPARATION 67

APPENDIX D. JAVA CODE FOR SIMKIT MODEL 75
A. MINE 76
B. MINE TYPE 77
C. POISSON MINEFIELD 78
D. RANDOM POINT GENERATOR 85
E. PROBABILITY COOKIE CUTTER SENSOR 87
F. PROBABILITY COOOKIE CUTTER MEDIATOR 88

G. SUBMARINE SMT 90
H. SUBMARINE LEAM 96
I. SIMULATE SUB MDA 102

LIST OF REFERENCES 107

INITIAL DISTRIBUTION LIST 111

LIST OF FIGURES

Figure 1. Different Mine Types.....2
Figure 2. Same Minefield Shown With Large or Small Standoff Distances. 4
Figure 3. Will Battlespace Preparation Improve P_{st}? 8
Figure 4. Square and Hexagon Configurations for Laying a Network upon a Minefield (Nodes Indicated by Heavy Dots). 9
Figure 5. Adjacent Nodes. 9
Figure 6. Node Numbering for a 5x5 nm Model.....13
Figure 7. Network for a 5x5 nm Model in Excel. 14
Figure 8. Generating Random 5x5 Grids. 14
Figure 9. Excel Implementation of Shortest Path Model. 16
Figure 10. Solver Parameters for Figure 5 Implementation. 16
Figure 11. Network for a 5x5 nm Model in GAMS. 17
Figure 12. Distance into Minefield (Sparse Versus Dense). 19
Figure 13. Distance into Minefield TDA with a Second Observation. 24
Figure 14. Shock Factor (After: Boerman, 1994, 20) 26
Figure 15. Minimum Standoff Distance Assuming Detection with Onboard Sonar (Not Drawn To Scale). 27
Figure 16. Nominal Values Assuming Detection with Onboard Sonar (Not Drawn To Scale). 28
Figure 17. Minimum Standoff Distance Assuming Avoidance of a Previously Reported Mine Position (Not Drawn To Scale).....28
Figure 18. Nominal Values Assuming Avoidance of a Previously Reported Mine Position (Not Drawn To Scale).....29
Figure 19. Calculating Minimum Standoff Distance Assuming Detection with Onboard Sensor (Not Drawn To Scale). 29
Figure 20. Calculating Minimum Standoff Distance Assuming Avoidance of a Previously Reported Mine Position (Not Drawn To Scale).....30
Figure 21. General Equation for Minimum Standoff Distance Assuming Detection. 30
Figure 22. General Equation for Minimum Standoff Distance Assuming Avoidance of a Previously Reported Mine Position. 31
Figure 23. Navigation Error Resulting in Mine Actuation.....36
Figure 24. SMT Model of Minefield Transit (From: Kim, 2002, 11). 39
Figure 25. Event Graph for SMT Model.....40
Figure 26. Incrementing Reentry Points for SMT Model. 41
Figure 27. MOAM Model of Minefield Transit (From: Kim, 2002, 47). 43
Figure 28. Event Graph for MOAM Model with Infinite Width Minefield.....43
Figure 29. Lateral Excursion Avoidance Method. 44
Figure 30. Differences in Models Choice of Avoidance Distance.....45
Figure 31. Event Graph for MOAM Model with Finite Width Minefield. 45
Figure 32. Event Graph for Baseline Lateral Excursion Avoidance Method. 46

Figure 33. Comparison of Different Obstacle Avoidance Models ("Mine Only" Case). 47
Figure 34. Comparison of Different Obstacle Avoidance Models ("Mine & NOMBO" Case). 48
Figure 35. Comparison of Different Obstacle Avoidance Models ("Higher Densities, Smaller Actuation Range" Case). 51
Figure 36. Submarine Experiencing Gridlock. 54
Figure 37. Gridlock Curves for Two Different Minimum Standoff Distances. 56

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. P_{st} Values for a 1 nm Arc Taken from Simulation (Nominal Values in Gray). 12
Table 2. Differences in Battlespace Preparation Model Implementation. 12
Table 3. Assigning Probability Distribution for NOMBOs (Gray Values Input By User). 14
Table 4. "First-Attempt" Evaluative TDA for Distance Into Minefield. 21
Table 5. Determining a Useful Estimate for How Far to Backup. 22
Table 6. Evaluative TDA for Distance Into Minefield. 22
Table 7. TDA Showing Range of Options Depending on Reliability of Input Criteria. 23
Table 8. Relationship between Event Graphs and Simkit (From: San Jose, 2001, 3). 37
Table 9. Comparison of Simulation and Analytical Mean (SMT "Mine Only" Case). 41
Table 10. Comparison of Simulation and Analytical Mean (SMT "Mine & NOMBO" Case). 42
Table 11. Comparison of LEAM Simulation and SMT, MOAM, and Upper Bound ("Mine Only" Case). 47
Table 12. Comparison of LEAM Simulation and SMT, MOAM, and Upper Bound ("Mine & NOMBO" Case). 48
Table 13. Parameters for Previous Scenarios 49
Table 14. NOMBO Clutter Categories 50
Table 15. Comparing New Parameters with Previous Scenarios 50
Table 16. Comparison of LEAM Simulation and SMT, MOAM, and Upper Bound ("Higher Densities, Smaller Actuation Range" Case). 50
Table 17. Gridlock Densities for 160-Yard Minimum Standoff Distance. 55
Table 18. Gridlock Densities for 450-Yard Minimum Standoff Distance. 55

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to acknowledge both Professors Steven E. Pilnick and Arnold H. Buss. Without their advice and long hours of cooperation, I could never have completed this project. I would also like to thank Patrick Madden of Johns Hopkins University Applied Physics Laboratory. His patience and interest in introducing me to the topic was crucial in my understanding. To the many other classmates and professors whose door I barged in on for consultation: thank you. Most of all, I would like to thank my wife, Flor, and son Liam for their love and support.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Since World War II, the United States Navy has had 14 ships hit by mines while suffering only two missile attacks. Although a U.S. submarine has not struck a mine since World War II, it is only a matter of time. With an increase in littoral operations, submarines can no longer ignore the mine threat. While minefield avoidance is intuitively preferable, a submarine could still encounter a minefield in two circumstances. First, a minefield may be an unavoidable barrier, such as mines laid across a chokepoint. Alternatively, as evidenced in Desert Storm, intelligence estimates of potentially mined waters can be wrong. In this second case, a submarine could stray into a minefield that it was specifically trying to avoid based on a priori knowledge. In either case, a submarine in a minefield desires to maximize its probability of safe transit using its high frequency sonar to detect mines. The task of mine detection and avoidance involves many tradeoffs. This thesis looked at five problems that could serve as a basis for tactical decision aids for submarine mine detection and avoidance.

First, bottom clutter along the ocean floor complicates the task of transiting through a minefield. A submarine's active sonar often detects bottom clutter, which can be anything from a rock outcropping to a rusty oil drum. These NOn-mine Mine-like Bottom Objects (NOMBOs) force the submarine to maneuver because there is no way to distinguish them from bottom mines. Spending more time and distance in the minefield by maneuvering increases the chance the submarine may strike an undetected mine. Given a constant mine density, areas with lower NOMBO density offer the best chance of safe transit since this minimizes submarine maneuvers. Without prior knowledge of the bottom topography of a mined chokepoint, a submarine would drive a straight-line course, thereby minimizing the time within the minefield. Battlespace preparation would involve the pre-survey of areas of interest to map locations of all NOMBOs. This map could help identify areas of high and low NOMBO densities. With this information, a submarine tasked with penetrating a minefield would have to choose between a short planned path through relatively high NOMBO density areas versus a longer planned path

through relatively lower NOMBO densities. This thesis uses a network model to prescribe the optimal route.

Second, when a submarine encounters a minefield, one tactic is to go around the field rather than attempt penetrating it. This tactic raises the question: if the submarine does encounter a mine, how far does the submarine need to back up in order to transit the edge of the minefield? In a dense minefield, the submarine commander would expect that he might not have to back up very far. The submarine probably encountered a mine near the edge. However, if the minefield is sparse, the first mine detected may be far into the minefield. This thesis provides information about distance into the minefield based upon assumptions of minefield distribution and density. Alternative courses of action for maneuvering are based upon assumptions from intelligence, battlespace preparation, bottom contours, other means of a priori knowledge, and assumptions about how mines are distributed in the minefield. This thesis distributes mines according to a spatial Poisson process. The submarine commander using this evaluative tactical decision aid can judge how sensitive the results are to the assumptions. For example, if a submarine with a 2000-yard sweep width detects a mine and based on intelligence assumes the minefield density is five mines per nm^2, the submarine should back up 1200 yards to get out of the minefield with probability 0.95.

Third, there are two situations where a submarine would avoid a mine contact. Either the submarine detected the mine with its onboard active sonar, or it is avoiding a reported mine position (detected by other assets or detected earlier by the submarine). In either case, the submarine must establish a minimum standoff distance at which it must order a maneuver to avoid the contact. Each possibility must take into account different factors, such as navigation error, sensor error, advance and transfer, and the mine's lethal range. This thesis generates a formula to compute minimum standoff distance for each of the two situations. For the case when detecting with an onboard sensor, minimum standoff distance equaled 160 yards (assuming 30 yard lethal range, 10 yard sensor error, 300 yard advance, and 300 yard transfer).

Fourth, the principal measure of effectiveness for a submarine conducting a minefield penetration is the probability of safe transit (P_{st}). With various mine densities and NOMBO densities, what is the chance that the submarine will safely transit across the minefield? The P_{st} depends upon the environment, the mines, and the submarine. A simulation in this thesis explores the interactions among these factors, and enables the study of alternate mine avoidance tactics. An obstacle avoidance algorithm, the Lateral Excursion Avoidance Method (LEAM), is developed and compared to two previously studied analytically simplified maneuvering models, as well as an analytical upper bound for P_{st}. Analysis of the simulation showed that the LEAM model improved P_{st} compared to the simpler models, and significantly, came close to the analytical upper bound.

Finally, if operators cannot distinguish between NOMBOs and mines, then the submarine must apply the same minimum standoff distance to all contacts. As the density level for NOMBOs and mines increases, a submarine will be less likely to find a safe path penetrating a minefield. This thesis terms this condition "gridlock." On any transit attempt, the submarine has three possibilities: safe transit, mine actuation, or gridlock. Using simulation, this thesis explores the density levels and minimum standoff distance that result in gridlock for a particular scenario. For example, in a five nm wide mined chokepoint, with a minimum standoff distance of 450 yards, a 0.50 probability of gridlock occurs at approximately 21 mines or NOMBOs per nm^2.

Having a submarine penetrate a minefield is a very risky concept of operations. This thesis builds the foundation for several tactical decision aids in mine detection and avoidance.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Future adversaries may have the means to render ineffective much of our current ability to project U.S. power overseas

2001 Quadrennial Defense Review discussing mine warfare

A. BACKGROUND

1. Naval Mine Warfare History

Naval mines were first deployed en masse in the U.S. Civil War, really caught the attention of the international community during the Russo-Japanese War (1904). These weapons of war are cheap, easy to deploy, difficult to detect, lethal, and will no doubt continue to be deployed. Since World War II, the United States Navy has had 14 ships hit by mines while suffering only two missile attacks.[1] Over the years, mines evolved from simple contact actuation to influence actuation upon pressure, magnetic, acoustic, or electric potential signatures. In addition to actuation methods, mines are also classified as either bottom, moored, or drifting. The 1907 Hague convention banned drifting mines. However, that did not stop Iraq from laying such mines during Desert Storm. In all, Iraq laid more than 1,300 mines in 1991.[2]

Although a U.S. submarine has not struck a mine since World War II, it is only a matter of time. The complex mine threat has not been fully addressed by submarines. For many years, the U.S. submarine force treated the mine threat lightly and declared that submarines were simply not going to operate near minefields. Unfortunately, friendly forces may not always know the location of a minefield. That is exactly the point when the enemy lays offensive minefields.

2. Availability of Mines

As of 1993, 15 nations produced and exported naval mines. These countries include China, Iraq, Libya, and North Korea. Eight other nations, including Iran, produced but did not export mines. Finally, 23 nations purchased but did not produce

[1] Robert J. Natter, "Access is Not Assured," U.S. Naval Institute Proceedings (January 2003): 39.

[2] Tony Perry, "Navy Faces a Terrorist Minefield in Persian Gulf," Los Angeles Times, 30 November 2002.

mines.[3] In the 1980's, the Soviets stockpiled over 200,000 mines. Now, Russia exports mines including rising propelled influence mines and bottom mines with up to 3,000 pounds of TNT equivalent.[4] Russia's rising propelled influence mines are moored mines that upon acquisition launch a torpedo or rocket for final homing. Some of these mines were specifically designed against submarines.

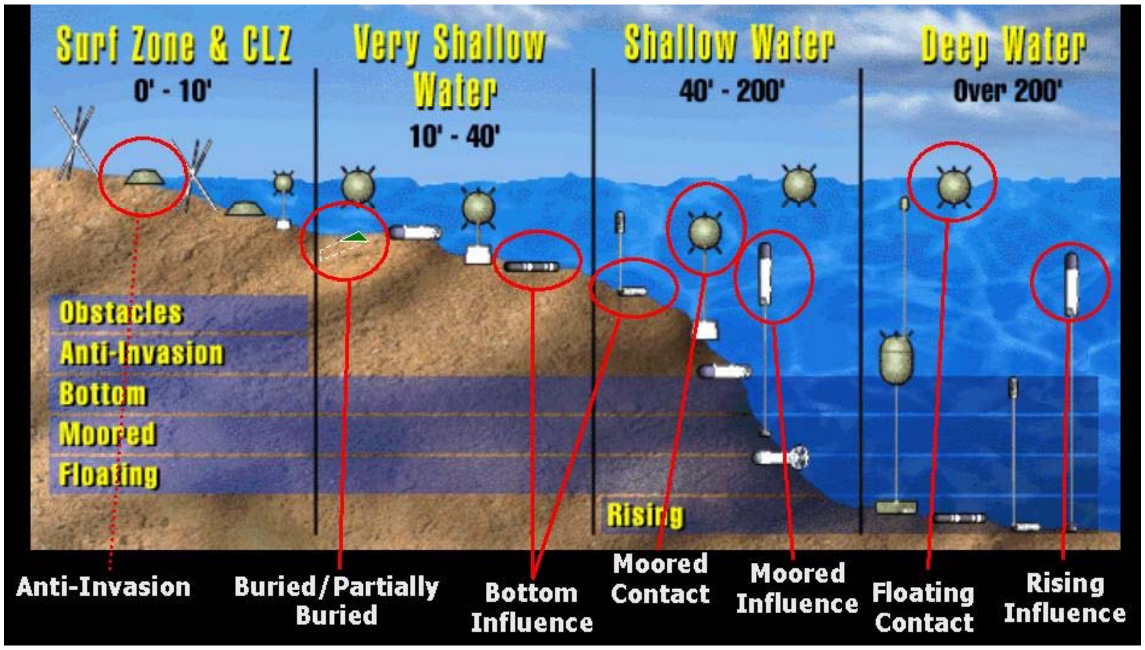


Figure 1. Different Mine Types.

3. Mine Detection and Avoidance

With an increase in operations in the littorals, submarines can no longer ignore the mine threat. While minefield avoidance is intuitively preferable, a submarine could still encounter a minefield in two circumstances. First, a minefield may be an unavoidable barrier, such as mines laid across a chokepoint. Alternatively, as evidenced in Desert Storm, intelligence estimates of potentially mined waters can be wrong. In this second case, a submarine could stray into a minefield that it was specifically trying to avoid based on a priori knowledge. In either case, a submarine in a minefield desires to maximize its probability of safe transit (P_{st}) using its high frequency sonar to detect mines. The task of mine detection and avoidance involves many tradeoffs.

[3] Johns Hopkins Applied Physics Laboratory. Need for and Utility of a Submarine Offboard Mine Search System (SOMSS). Laurel, MD. (30 August 1993), 7.

[4] Ibid: 10.

First, if a submarine detects a mine, how should it best maneuver to maximize P_{st}? Maneuvering around detected contacts with as wide of a clearance as possible will improve the submarine's chance of survival versus the detected contact. However, any maneuver will also increase the distance and time spent within the minefield. This will increase the chance the submarine may strike an undetected mine. Finding the proper balance between these two competing ideas requires knowledge of: submarine maneuvering characteristics, detection capabilities, and minefield characteristics.

Secondly, at what depth should a submarine transit through a minefield? Submarines do a fairly good job of detecting moored mines regardless of search depth. An object on high frequency sonar that appears floating in the water column, in a fixed point in space, is likely a mine. Detecting bottom mines is much more challenging because other objects on the ocean floor may be difficult to distinguish from mines. These NOMBOs (NOn-mine Mine-like Bottom Objects) significantly hamper a submarines search for bottom mines. Operating close to the bottom increases the submarine's probability of detection (P_d) against bottom mines. However, operating deep also increases the potential damage of any bottom mine that explodes near the submarine.

Finally, a submarine must establish some minimum standoff distance for all the contacts that it detects. This minimum standoff distance should account for the lethal range of the mine, advance and transfer, set and drift, and other detection delays. Hence, this minimum standoff distance will vary for different warhead sizes. The submarine would like to make this radius as large as possible for safety considerations. However, if made too large, it becomes impossible to maneuver within a minefield. When mine and NOMBO densities are too high for a ship to penetrate the minefield, this is termed "gridlock." Obviously, the size of the minimum standoff distance will affect the densities necessary for gridlock, as demonstrated below in Figure 2. The minefield on the left has a large standoff distance and the submarine encounters "gridlock." The minefield on the right has a smaller standoff distance and the submarine can penetrate the minefield.

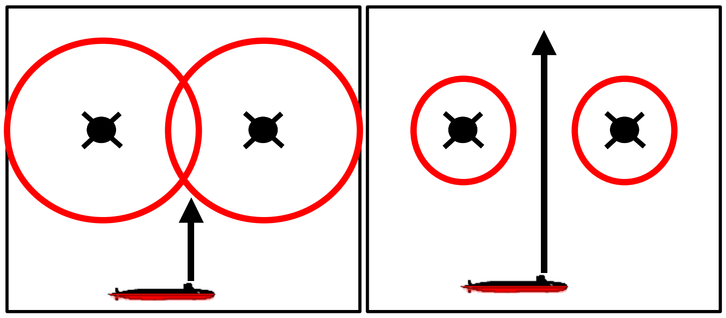


Figure 2. Same Minefield Shown With Large or Small Standoff Distances.

4. Previous Studies

In May 2001, the director of the Submarine Warfare Division (N77) directed the formation of a Mine Countermeasures Technical Advisory Group (MCMTAG). One of their purposes is to gain a better understanding of the factors affecting submarine mine detection and avoidance. In 2001, a Submarine Mine Detection and Avoidance Capability Assessment produced several recommendations. With limited data from at-sea events, the assessment did not cover all factors affecting Mine Detection and Avoidance (MDA). The MCMTAG concluded that modeling and simulation are necessary since further at-sea events are limited by fiscal and time constraints. The Center for Naval Analyses (CNA) prepared two earlier MDA reports using simulation for the Commander Submarine Squadron Twelve in the early 1990's. However, a number of simplifying assumptions warrant revisiting since submarines must now operate more in the littorals. Specific assumptions and inputs that are newly addressed include:

- Shallow water operations (less than 300 feet)
- Mixed minefield composition (both bottom and moored mines)
- Increased NOMBO density

In December 2002, Chihoon Kim's thesis at the Naval Postgraduate School had a result that can be counterintuitive at first glance. Given some high frequency sonar's probability of detection (P_d) for mines, it is sometimes better for a submarine to turn off its sonar and enter blindly into the minefield. Why would you turn off your sensor? With a high rate of NOMBOs and a low probability of detection, the submarine will end up spending too much time and distance in the minefield as it dodges around one false contact to another. Kim's thesis emphasized that his second more detailed model was more optimistic for P_{st} and suggested that enhanced transit tactics are worthy of investigation. A number of his assumptions warrant enhancement, such as: modeling advance and transfer, including reaction time, realistic mine actuation ranges, and higher mine and NOMBO densities. In addition, his simulation model assumed that the mines appeared as a spatial Poisson process.[5] Appendix B provides background on the spatial Poisson process.

Many recent mine detection and avoidance studies have addressed the use of unmanned undersea vehicles (UUV's). UUV's will search the minefield and give the submarine a map of the minefield before entry. After using information from the UUV, a submarine would still use active sonar when transiting the UUV's safe route. The submarine may still have to conduct MDA for mines along its path that the UUV did not detect. This study addresses only the submarine and its onboard sensors.

Another research area integral to the modeling of submarine mine detection and avoidance is "obstacle avoidance" or "path planning." The robotics field heavily researches both of these topics. Path planning is a general research area where objects maneuver through a path of obstacles. The field of study divides path planning into two types of planners: reactive and predictive.

In predictive path planning, the algorithm knows the obstacle position a priori and the goal is to plan the best path (usually the shortest) through the given obstacles. One paper describes a planner as predictive if "it must generate a sequence of steps before it can determine any one step in the sequence."[6] A submarine could use this type of planning when developing a route through a minefield that a UUV has pre-surveyed. A submarine following a UUV's path that runs across a mine previously undetected must create a new path around that object.

[5] Chihoon Kim, "The Effect of Sensor Performance on Safe Minefield Transit" M.S. Thesis, Naval Postgraduate School: 12.

[6] Tychonievich, Lou, Thomas C. Smith, and Robert Evans, "Influence Networks: A Reactive Planning Architecture." Proceedings Seventh IEEE Conference on Artificial Intelligence for Applications, 1991: 354.

Reactive planning is “able to generate its next step without having to generate any subsequent steps.”[7] The concept of gridlock is particularly familiar in reactive path planning. Because reactive path planning is “inherently short-term in scope,”[8] the path can lead to the vehicle becoming “trapped in certain boxed canyon situations.”[9] This boxed canyon concept is the scenario described here as “gridlock.” A submarine penetrating a minefield while relying solely on its onboard sensor would fall into the category of reactive path planning.

B. OBJECTIVES

This study has five objectives. Each concept will provide a basis for a tactical decision aid for a submarine conducting MDA. Will battlespace preparation provide an advantage when conducting a minefield penetration? How far into a minefield is the submarine once it has detected its first mine? What minimum standoff distance should the submarine keep from mines for a given warhead size? What is the probability of safe transit for varying sonar performance and minefield compositions? At what level of contact density should the submarine turn around?

Chapter II answers the first question using a network model. Chapter III discusses the next question with an analytical model and an emphasis on simplifying assumptions. Calculations of minimum standoff distance in Chapter IV serve as inputs to a simulation. A simulation in Chapters V through VI investigates the final two questions.

[7] Tychonievich, Lou, Thomas C. Smith, and Robert Evans, “Influence Networks: A Reactive Planning Architecture.” Proceedings Seventh IEEE Conference on Artificial Intelligence for Applications, 1991: 354.

[8] Ibid.

[9] Hyland, John C. and Fred J. Taylor, “Mine Avoidance Techniques for Underwater Vehicles,” IEEE Journal of Oceanic Engineering, Vol. 18, NO. 3 (July 1993): 343.

II. BATTLESPACE PREPARATION

Our battlegroups and Joint Task Force Commanders need an easily interpretable undersea battlespace picture that depicts bathymetry, environmental effects on weapon and sonar performance as well as mine and undersea vehicle threats.

VADM Grossenbacher, COMSUBLANT (USW Fall 2000)

A. PROBLEM DEFINITION

Many factors affect the probability of safe transit (P_{st}) for a submarine transiting through mineable waters. The existence of false contacts for bottom mines complicates the task. A submarine's active sonar often detects bottom clutter, which can be anything from a rock outcropping to a rusty oil drum. These NOn-mine Mine-like Bottom Objects (NOMBOs) force the submarine to maneuver because there is no way to distinguish them from bottom mines. Spending more time and distance in the minefield by maneuvering increases the chance the submarine may strike an undetected mine. Given a constant mine density, areas with lower NOMBO density offer the best chance of safe transit since this minimizes submarine maneuvers.

The MDA problem is very important when a submarine crosses a shallow-water region with areas of high NOMBO density. Without the ability to discriminate between actual mines and NOMBOs, the submarine must avoid NOMBOs just as it avoids mines. In the MDA problem, battlespace preparation would involve the pre-survey of areas of interest to map locations of all NOMBOs. This map could help identify areas of high and low NOMBO densities. With this information, a submarine tasked with penetrating a minefield would avoid regions of high NOMBO density to increase its P_{st}.

In addition, by using bathymetric data a submarine will have an estimate of sonar performance in the area of interest. Different ocean areas will have varying levels of sonar performance. This sonar performance can be represented by an estimate of probability of detection (P_d) against mines, yielding average probability of detecting a mine as another useful metric.

The U.S. Navy is taking a keen interest in the idea of battlespace preparation, particularly when it comes to MCM. The eventual goal is that:

> Battlespace preparation efforts performed (weeks or months) prior to the commitment of MCM forces will substantially reduce the time and effort required to conduct a successful MCM campaign. Mine countermeasures battlespace preparation includes the establishment of bottom mapping, survey, and intelligence databases for use in determining such as … location and density of significant mine-like bottom objects, and bottom clutter characteristics of a geographic region.[10]

Without battlespace preparation, a submarine would drive a straight-line course across a mined chokepoint, thereby minimizing the time within the minefield. With battlespace preparation, a new route of low NOMBO densities may increase this path length.

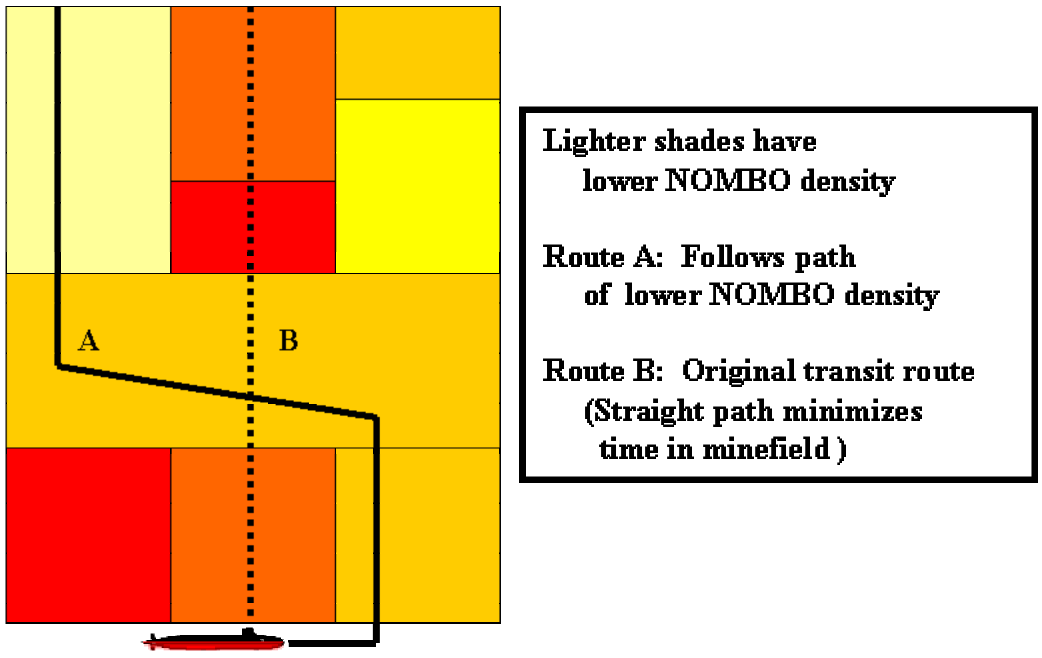

A
B
Lighter shades have lower NOMBO density
Route A: Follows path of lower NOMBO density
Route B: Original transit route (Straight path minimizes time in minefield)

Figure 3. Will Battlespace Preparation Improve P_{st}?

B. ANALYTICAL MODEL

Solving a shortest path network problem can determine the route with the highest probability of safe transit. The model assumes battlespace preparation records the location of NOMBOs in an area of interest. This large area can be broken up into arbitrarily small sections of different densities. The model considers transit between these small areas to take place across nodes. The specific implementation of this model would determine the particular breakup of the region into smaller areas and the

[10] U.S. Navy Department. AN/BLQ-11 Long Term Mine Reconnaissance System (LMRS). NWP 3-15.5 (Draft): G8.

assignment of nodes. In other words, one could break up the chokepoint into any number of configurations as shown below in Figure 4.

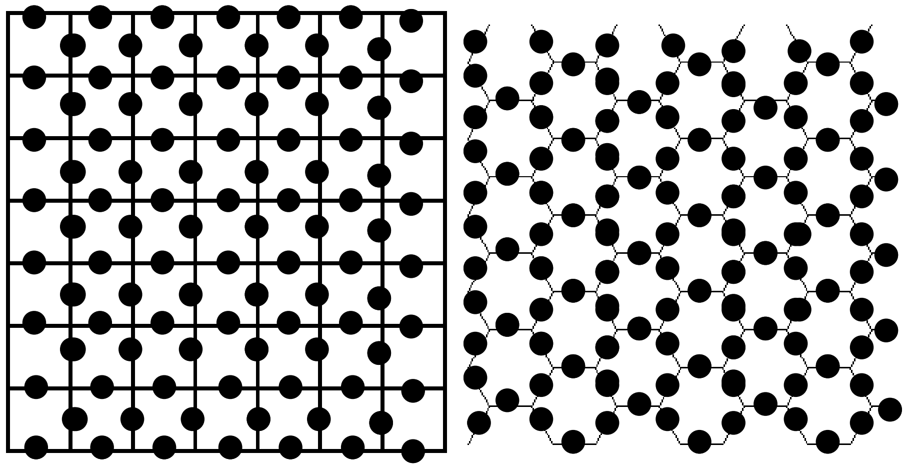


Figure 4. Square and Hexagon Configurations for Laying a Network upon a Minefield (Nodes Indicated by Heavy Dots).

For example, assume the two configurations above are for a region 7 nm by 7 nm with n-1 nodes. Assume that the submarine can enter anywhere along the bottom and exit anywhere at the top. The model requires two dummy nodes (node 0 for the start and node n for the finish). The submarine can travel between any adjacent nodes. The specific implementation of the model determines which nodes to label as "adjacent." Figure 5 shows one possible definition of "adjacent" nodes.

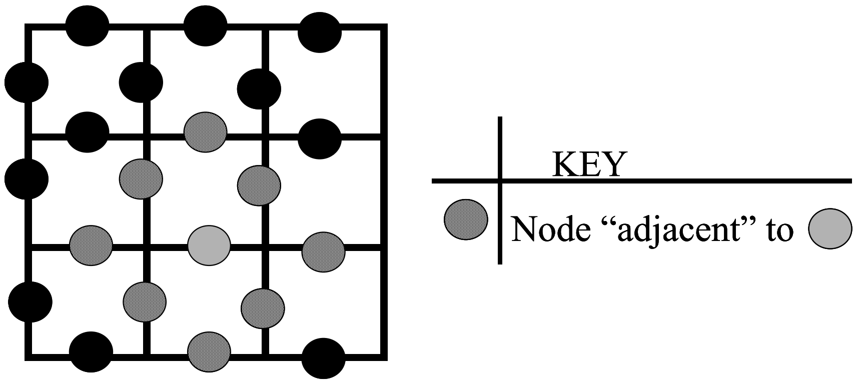


Figure 5. Adjacent Nodes.

C. MODEL FORMULATION

1. Non Linear Program

Regardless of the particular node network chosen, the model formulation remains the same.

Indices

i = index for source node (i = 0,1,…n-1) (note: node n is never a source)

j = index for destination node (j = 1,…n) (note: node 0 is never a destination)

Parameters

P_{ij} = the probability of actuating a mine from node i to node j. $0 \le P_{ij} \le 1$

Variables

Y_{ij} = binary decision variable for transit of arc from node i to adjacent node j.

Constraints

$$\sum_{j:(i,j)\in A} Y_{ij} - \sum_{j:(j,i)\in A} Y_{ji} = \begin{cases} -1 & i = 0 \\ 0 & i = [1, n-1] \\ 1 & i = n \end{cases}$$

where $Y_{ij} \in \{0,1\}$

A = set of adjacent nodes

Objective Function

$$Max \prod_{(i,j)\in A} \left(1 - P_{ij}\right) Y_{ij}$$

Summary of Constraints and Objective Function

The balance of flow constraints ensure that one unit leaves the start node (node 0) and that one unit arrives at the exit node (node n). The remaining balance of flow constraints (nodes 1 to n-1) ensures that any unit entering a particular node will also exit the same node. In other words, the submarine will transit all the way across the minefield. The objective function is maximizing the probability of safe transit for the chosen path. As written, the objective function is nonlinear since it multiplies the decision variables (weighted by the probability of safe transit on each arc).

2. Linear Program

Converting the above formulation into a linear program has important benefits. A linear program is much easier to solve. As mentioned, the only aspect of the previous formulation that is nonlinear is the objective function. The rewritten objective function is:

$$\prod_{(i,j)\in A}\left(1-P_{ij}\right)Y_{ij} = e^{\log\left(\prod_{(i,j)\in A}\left(1-P_{ij}\right)Y_{ij}\right)} = e^{\sum_{(i,j)\in A}\log\left(\left(1-P_{ij}\right)Y_{ij}\right)}$$

Now, since the exponential function is monotonic increasing, maximizing $e^{\sum_{(i,j)\in A}\log\left(\left(1-P_{ij}\right)Y_{ij}\right)}$ is equivalent to maximizing $\sum_{(i,j)\in A}\left[\log(1-P_{ij})\right]Y_{ij}$.

Although maximizing the sum of log probabilities is less intuitive, it is mathematically equivalent to maximizing the product of the probabilities. The new linear objective function for finding an optimal path is shown below. The indices, parameters, variables, and constraints remain the same.

Objective Function

$Max \sum_{(i,j)\in A}\left[\log(1-P_{ij})\right]Y_{ij}$ for all i, j nodes that are adjacent

D. DETERMING PROBABILITY OF SAFE TRANSIT

The model takes the values for probability of safe transit for each arc from simulation. Simulation runs from Chapter V have a fixed NOMBO density, mine density, and P_d. The battlespace preparation model includes the mean P_{st} from these minefield transits as shown in Table 1. Then, the battlespace preparation model uses each node's characteristics (NOMBO density, mine density, P_d, and transit length) to assign the P_{st} for a particular arc.

Probability of Safe Transit for 1 nm transit with given NOMBO and Mine Densities	Low	Medium	High	
	2	4	6	Mine Density
	Pst	Pst	Pst	
0-1 NOMBOs/sq nm	0.950	0.890	0.840	
1-2 NOMBOs/sq nm	0.925	0.865	0.815	
2-3 NOMBOs/sq nm	0.900	0.840	0.790	
3-4 NOMBOs/sq nm	0.875	0.815	0.765	
4-5 NOMBOs/sq nm	0.850	0.790	0.740	
5-6 NOMBOs/sq nm	0.825	0.765	0.715	
6-7 NOMBOs/sq nm	0.800	0.740	0.690	
7-8 NOMBOs/sq nm	0.775	0.715	0.665	
8-9 NOMBOs/sq nm	0.750	0.690	0.640	
9-10 NOMBOs/sq nm	0.725	0.665	0.615	
11-12 NOMBOs/sq nm	0.700	0.640	0.590	
12-13 NOMBOs/sq nm	0.675	0.615	0.565	
13-14 NOMBOs/sq nm	0.650	0.590	0.540	
14-15 NOMBOs/sq nm	0.625	0.565	0.515	

Table 1. P_{st} Values for a 1 nm Arc Taken from Simulation (Nominal Values in Gray).

E. EXAMPLE MODEL IMPLEMENTATIONS

Both Excel and GAMS (General Algebraic Modeling System) implement the following example. The Excel example solves the nonlinear formulation while the GAMS example solves the linear formulation. Both examples deal with a 5 x 5 nm area, but they each have a different network layout. Either software could solve other generated examples. For example, a GAMS model implementation could solve either network layout, simulation or equation-generated P_{st}, and solve a linear or nonlinear objective function.

	Excel Example	**GAMS Example**
Objective Function	Non Linear	Linear
Network	26 Nodes & 86 Arcs	52 Nodes & 115 Arcs

Table 2. Differences in Battlespace Preparation Model Implementation.

The Excel implementation is flexible and offers a solution to quickly generated different scenarios. GAMS can handle larger networks that better represent the scale of problems needed to solve real-world battlespace preparation networks.

1. Excel Implementation

The Excel example of the model assumes that the survey records NOMBO densities in blocks measuring one nm^2. One nm^2 survey data is very optimistic and explores the limits of future surveying efforts. The model assigns the center of each survey block as the node in the network. Each node has an associated NOMBO density and a probability of detection (P_d). The model uses the same P_d for both mines and NOMBOs. The model can then assign probability of safe transit (P_{st}) to each arc based upon the associated NOMBO density, P_d, mine density, and transit length. The P_{st} value for each arc is determined from a table generated by simulation.

The example uses a 5 x 5 nm area consisting of twenty-five 1-nm^2 blocks. The submarine transits from bottom to top (or South to North) and cannot go around the minefield (it is a narrow chokepoint). Node numbers 0 and 26 are the dummy entry and exit nodes discussed earlier.

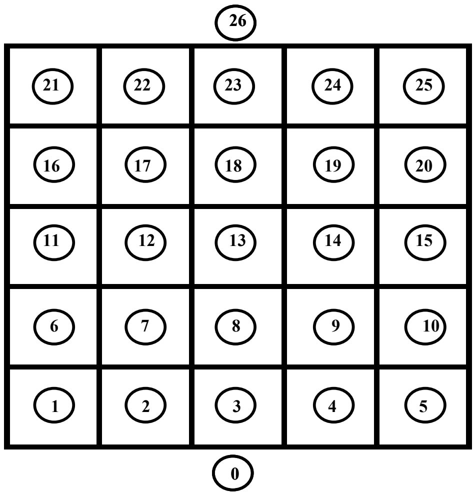


Figure 6. Node Numbering for a 5x5 nm Model.

A submarine can move between adjacent blocks. Adjacent blocks include diagonal moves between blocks. This small example contains only 86 arcs as shown below in Figure 7. There is no arc between nodes 1 and 2 because the shortest path would have chosen to go directly from node 0 to 2. The same reasoning explains the other "missing" arcs in the first and last row.

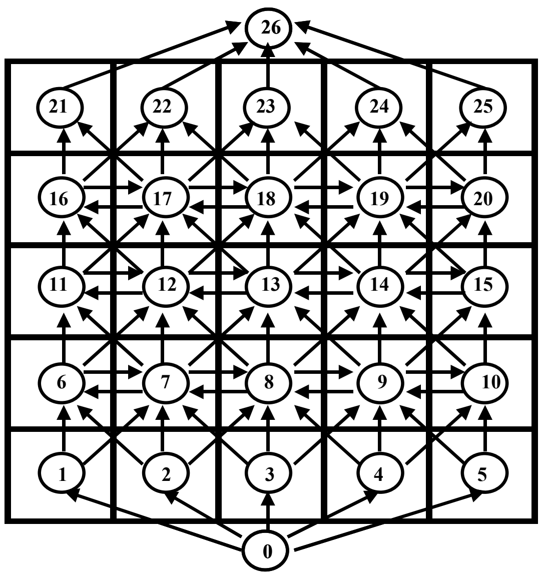


Figure 7. Network for a 5x5 nm Model in Excel.

The spreadsheet implementation of the model has one portion that can generate 5x5 nm grids with random NOMBO densities for each block. The distribution of NOMBO densities is input by the user as shown in Table 3. Excel displays the grid graphically with color-coding and the actual NOMBO densities are in a table as shown in Figure 8.

NOMBO densities (inclusive)		
Lower Value	Upper Value	Prob. Occurrence
0	**1**	0.2
1	**2**	0.2
2	**3**	0.2
3	**4**	0.1
4	**5**	0.05
5	**6**	0.05
6	**7**	0.05
7	**8**	0.05
8	**9**	0.05
9	**10**	0.04
10	**11**	0.01
11	**12**	0
12	**13**	0
13	**14**	0
14	**15**	0

Table 3. Assigning Probability Distribution for NOMBOs (Gray Values Input By User).

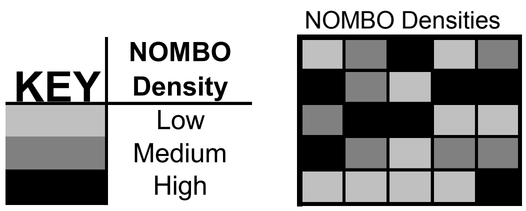


Figure 8. Generating Random 5x5 Grids.

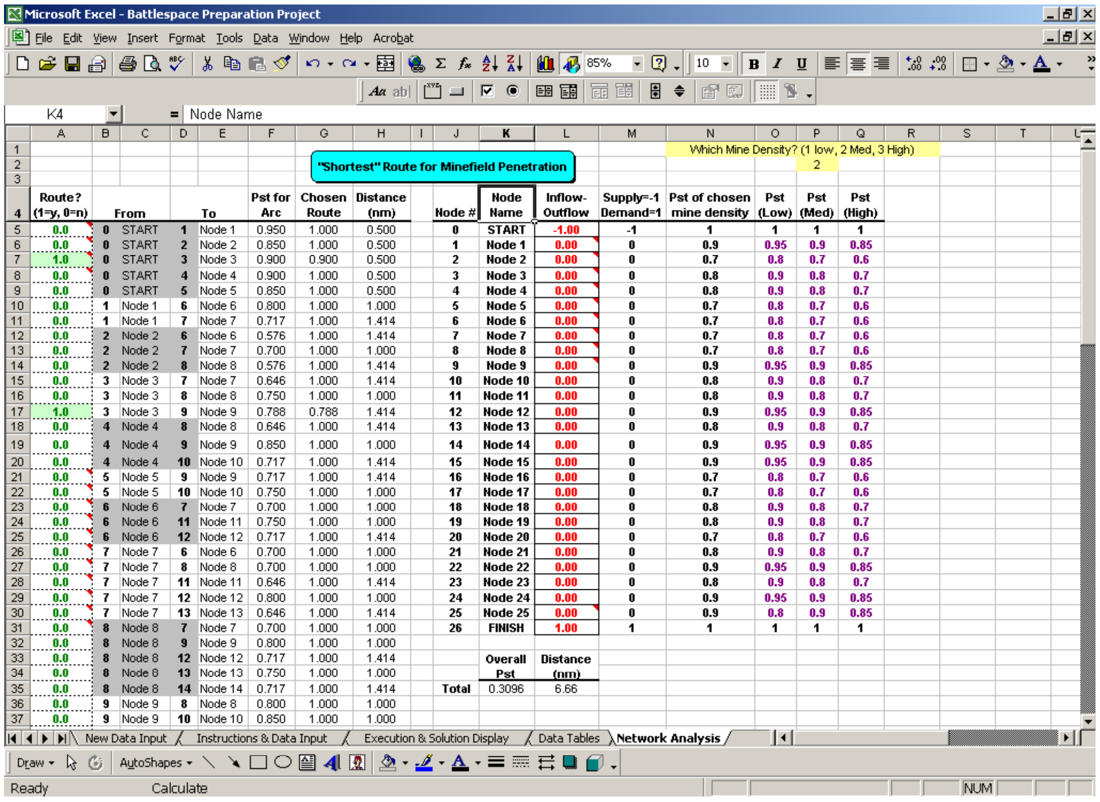
In addition to NOMBO densities, a submarine might also have data concerning sonar performance in the region of interest. Environmental conditions may result in different probabilities of detection (P_d) against mines in different ocean areas. This model maps random P_d's onto the 5 x 5 nm grid, so each block has an associated P_{st} and P_d. Excel displays the grid graphically with color-coding similar to Figure 8. The Pd distribution comes from a table similar to Table 3. The P_{st} in the Excel example uses the tabulated values from Chapter V's simulation.

Implementing the model in Excel solves the nonlinear network model as shown in Figure 9. Column A is the decision variable Y_{ij}. Columns N and O show the balance of flow constraints. Column F calculates the P_{st} for each arc.

Using the Solver tool as shown in Figure 8, we can choose our objective function as P_{st} (cell K35). The Y_{ij} decision variables (cells A5:A85) are the variable cells. The balance of flow constraint ensures that inflow-outflow (cells L5:L31) equals supply/demand (cells M5:M31). Besides balance of flow, the only other constraint is that Y_{ij} must be binary.

2. Excel Limitations

Using Excel to analyze this problem rather than specialized optimization software has several advantages: it provides a quick answer, it is easy to produce graphical output that is useful to the decision maker, and the software is almost universally available. However, there is a significant pitfall to Excel, regarding the solver. Excel's solver is limited to 200 decision variables, which is equivalent to the number of arcs in a network flow problem. This limitation is not a factor for the 5x5 grid examined in this study, which has 86 arcs. Consider, however, a 10x10 grid, with 416 arcs (decision variables) or a 25x25 grid, with 3496 arcs (decision variables), and one soon realizes that Excel is not the tool of choice for real world problems of this type.



Microsoft Excel - Battlespace Preparation Project

K4 = Node Name

"Shortest" Route for Minefield Penetration

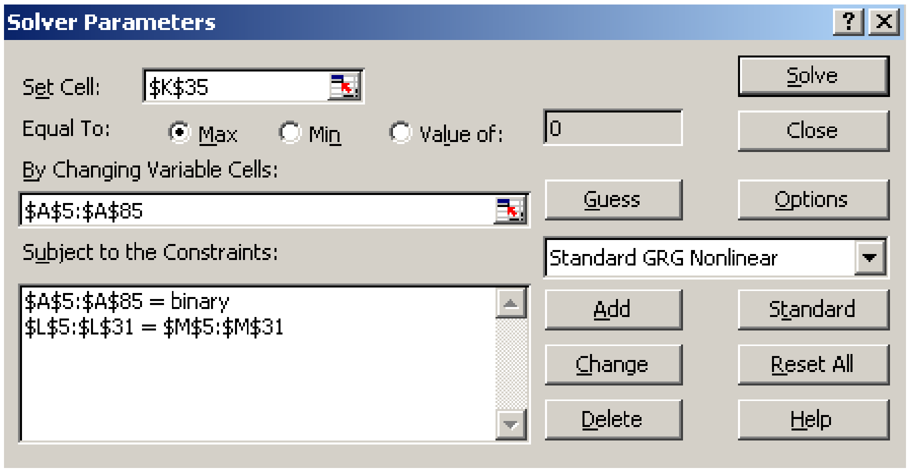
Which Mine Density? (1 low, 2 Med, 3 High) 2

	Route? (1=y, 0=n)	From		To		Pst for Arc	Chosen Route	Distance (nm)	Node #	Node Name	Inflow-Outflow	Supply=-1 Demand=1	Pst of chosen mine density	Pst (Low)	Pst (Med)	Pst (High)
5	0.0	0	START	1	Node 1	0.950	1.000	0.500	0	START	-1.00	-1	1	1	1	1
6	0.0	0	START	2	Node 2	0.850	1.000	0.500	1	Node 1	0.00	0	0.9	0.95	0.9	0.85
7	1.0	0	START	3	Node 3	0.900	0.900	0.500	2	Node 2	0.00	0	0.7	0.8	0.7	0.6
8	0.0	0	START	4	Node 4	0.900	1.000	0.500	3	Node 3	0.00	0	0.8	0.9	0.8	0.7
9	0.0	0	START	5	Node 5	0.850	1.000	0.500	4	Node 4	0.00	0	0.8	0.9	0.8	0.7
10	0.0	1	Node 1	6	Node 6	0.800	1.000	1.000	5	Node 5	0.00	0	0.7	0.8	0.7	0.6
11	0.0	1	Node 1	7	Node 7	0.717	1.000	1.414	6	Node 6	0.00	0	0.7	0.8	0.7	0.6
12	0.0	2	Node 2	6	Node 6	0.576	1.000	1.414	7	Node 7	0.00	0	0.7	0.8	0.7	0.6
13	0.0	2	Node 2	7	Node 7	0.700	1.000	1.000	8	Node 8	0.00	0	0.7	0.8	0.7	0.6
14	0.0	2	Node 2	8	Node 8	0.576	1.000	1.414	9	Node 9	0.00	0	0.9	0.95	0.9	0.85
15	0.0	3	Node 3	7	Node 7	0.646	1.000	1.414	10	Node 10	0.00	0	0.8	0.9	0.8	0.7
16	0.0	3	Node 3	8	Node 8	0.750	1.000	1.000	11	Node 11	0.00	0	0.8	0.9	0.8	0.7
17	1.0	3	Node 3	9	Node 9	0.788	0.788	1.414	12	Node 12	0.00	0	0.9	0.95	0.9	0.85
18	0.0	4	Node 4	8	Node 8	0.646	1.000	1.414	13	Node 13	0.00	0	0.8	0.9	0.8	0.7
19	0.0	4	Node 4	9	Node 9	0.850	1.000	1.000	14	Node 14	0.00	0	0.9	0.95	0.9	0.85
20	0.0	4	Node 4	10	Node 10	0.717	1.000	1.414	15	Node 15	0.00	0	0.9	0.95	0.9	0.85
21	0.0	5	Node 5	9	Node 9	0.717	1.000	1.414	16	Node 16	0.00	0	0.7	0.8	0.7	0.6
22	0.0	5	Node 5	10	Node 10	0.750	1.000	1.000	17	Node 17	0.00	0	0.7	0.8	0.7	0.6
23	0.0	6	Node 6	7	Node 7	0.700	1.000	1.000	18	Node 18	0.00	0	0.8	0.9	0.8	0.7
24	0.0	6	Node 6	11	Node 11	0.750	1.000	1.000	19	Node 19	0.00	0	0.8	0.9	0.8	0.7
25	0.0	6	Node 6	12	Node 12	0.717	1.000	1.414	20	Node 20	0.00	0	0.7	0.8	0.7	0.6
26	0.0	7	Node 7	6	Node 6	0.700	1.000	1.000	21	Node 21	0.00	0	0.8	0.9	0.8	0.7
27	0.0	7	Node 7	8	Node 8	0.700	1.000	1.000	22	Node 22	0.00	0	0.9	0.95	0.9	0.85
28	0.0	7	Node 7	11	Node 11	0.646	1.000	1.414	23	Node 23	0.00	0	0.8	0.9	0.8	0.7
29	0.0	7	Node 7	12	Node 12	0.800	1.000	1.000	24	Node 24	0.00	0	0.9	0.95	0.9	0.85
30	0.0	7	Node 7	13	Node 13	0.646	1.000	1.414	25	Node 25	0.00	0	0.9	0.8	0.9	0.85
31	0.0	8	Node 8	7	Node 7	0.700	1.000	1.000	26	FINISH	1.00	1	1	1	1	1
32	0.0	8	Node 8	9	Node 9	0.800	1.000	1.000								
33	0.0	8	Node 8	12	Node 12	0.717	1.000	1.414		Overall Pst	Distance (nm)					
34	0.0	8	Node 8	13	Node 13	0.750	1.000	1.000								
35	0.0	8	Node 8	14	Node 14	0.717	1.000	1.414	Total	0.3096	6.66					
36	0.0	9	Node 9	8	Node 8	0.800	1.000	1.000								
37	0.0	9	Node 9	10	Node 10	0.850	1.000	1.000								

New Data Input / Instructions & Data Input / Execution & Solution Display / Data Tables / Network Analysis

Ready Calculate NUM

Figure 9. Excel Implementation of Shortest Path Model.



Solver Parameters

Set Cell: $K$35

Equal To: [✓ Max] Min Value of: 0

By Changing Variable Cells: $A$5:$A$85

Subject to the Constraints:

$A$5:$A$85 = binary
$L$5:$L$31 = $M$5:$M$31

Standard GRG Nonlinear

Figure 10. Solver Parameters for Figure 5 Implementation.

3. GAMS Implementation

As mentioned earlier, GAMS is a much more powerful optimization tool than the solver packaged with Excel. In this example, the model assumes the same block size, node information (Pd and NOMBO density) and P_{st} assigned to each arc. The problem's formulation uses a linear objective function. Another important difference is that this model has a different network layout as shown below in Figure 11. This network has 52 nodes (including the two dummy nodes) and 115 arcs. These differences show the flexibility of using a nodal network model. The code for the GAMS implementation is included in Appendix C.

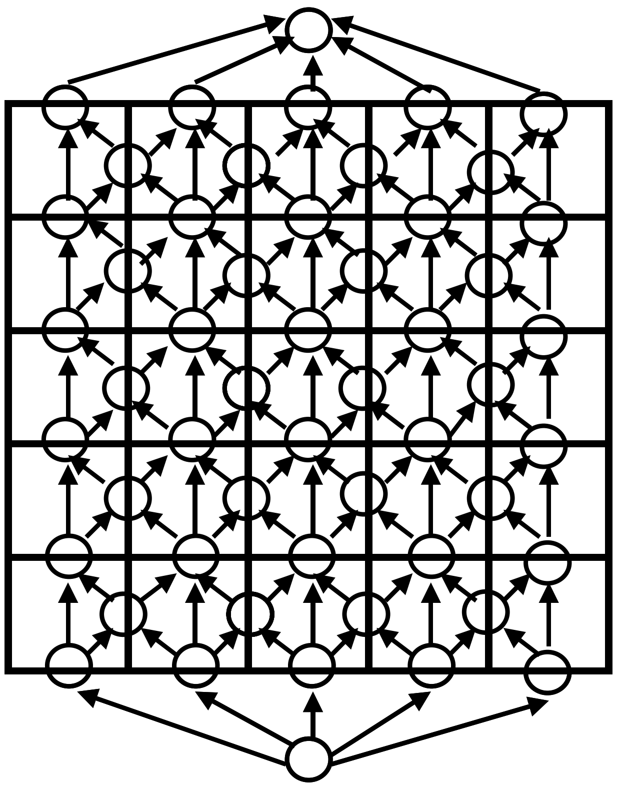


Figure 11. Network for a 5x5 nm Model in GAMS.

G. BENEFITS OF MODEL

Solving a network shortest path shows that the concept of battlespace preparation for minefield penetration is more difficult than it appears at first glance. The Excel model shows a way to quickly implement and analyze this problem in a small-scale network. Demonstrating this model with software familiar to most decision makers makes this implementation advantageous. Multiple scenarios run quickly and Excel's graphics capabilities enhance the display of input and output. However, if actually applied to a realistic problem (a 25 x 25 nm grid), the network problem exceeds Excel's solver capacity and more powerful optimization software is needed (such as General Algebraic Modeling System (GAMS)).

THIS PAGE INTENTIONALLY LEFT BLANK

III. DISTANCE INTO MINEFIELD

The random minefield assumption is robust to the kinds of deviations from the ideal of regularity that actually occur in practice. It may not be true that mines are *deliberately* placed at random, but the effect is much the same.

Daniel H. Wagner, ed. Naval Operations Analysis

A. PROBLEM DEFINITION

When a submarine encounters a minefield, one tactic is to skirt around the edge of the minefield and pass around it rather than attempt penetrating the minefield. After encountering the first mine, how far does the submarine need to back up in order to safely transit the edge of the minefield? Is 200 yards or 2000 yards going to be enough?

In a dense minefield, the submarine commander would expect that he might not have to back up very far. The submarine probably encountered a mine near the edge. However, if the minefield is sparse, the first mine detected may be far into the minefield (see Figure 12). Analytical evaluation provides information about distance into the minefield based upon assumptions of minefield distribution and density. Alternative courses of action for maneuvering are based upon assumptions from intelligence, battlespace preparation, bottom contours or other means of a priori knowledge. The submarine commander using this evaluative tactical decision aid (TDA) can judge how sensitive the results are to the assumptions.

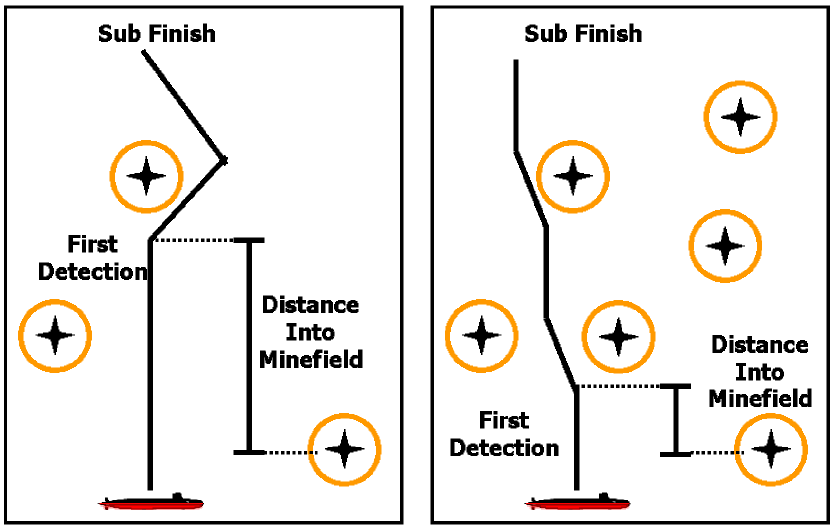


Figure 12. Distance into Minefield (Sparse Versus Dense).

B. ASSUMPTIONS

Assume that the uncertain number of functioning mines is modeled with the Poisson distribution. Further, assume that an uncertain number of functioning mines are distributed uniformly over a geographic area is equivalent to assuming that the mines are distributed according to a spatial Poisson process. For example, a spatial Poisson process could generate 100 mines in a 10 x 10 nm minefield. The Poisson process could have generated 88, 95, 103, or 140 mines in the minefield. In each case, after the number of mines is determined, say at 100 in the first case, it would be equivalent to distributing the mines according to a Uniform distribution. See Appendix B for more details of a Spatial Poisson process.

C. ANALYTICAL MODEL

Now, if the mines end up in a random pattern according to a spatial Poisson process, analytical computations can be made. The concept of "distance into a minefield" is related to "mean free paths" of Section 9.3 of Washburn's Search and Detection. In this section, Washburn asks, "How far will a marble of diameter W [w] roll in a Poisson field with density d [δ] before it (more precisely, its shadow) encounters a target?"[11]

1. Variables

w = the sweep width

X = distance to the first mine (the free path)

x = distance the sub has traveled

δ = density of minefield (mines per unit area)

2. Exponential Distribution

Given the above variables, P(X>x) = $e^{-\delta \mathrm{w} \mathrm{x}}$. Therefore, X is an exponential random variable with parameter $\lambda = \delta \mathrm{w}$. The mean of an exponential distribution is E(X) = $\frac{1}{\lambda} = \frac{1}{\delta \mathrm{w}}$. Therefore, the expected distance to the first mine, E(X), is equal to

[11] Alan R. Washburn, Search and Detection, 3rd Ed. (Institute for Operations Research and the Management Sciences, 1996), 9-4.

$\frac{1}{\delta \mathrm{w}}$. For example, if a sub with a one nm sweep width enters a minefield with 5 mines/nm^2 then δw = 5 mines/nm. E(X) = 1/5 nm which simply means that with mines occurring on average at 5 times every nm, the long run average distance to the first mine encounter is 1/5 of a nm, or 400 yards. Of course, on any particular attempt, the first mine may be encountered at any distance into the minefield (less than or equal to x) with some probability given by $1-e^{-\delta \mathrm{wx}}$.

Applying the equation E(X) $=\frac{1}{\delta \mathrm{w}}$ results in the first-attempt of an evaluative TDA appearing below in Table 4.

Sweep Width (yards)	Minefield Density (mines per square nm)	1	2	3	4	5	6	7	8	9	10
4000	Distance into Minefield (yards)	1000	500	333	250	200	167	143	125	111	100
3500	Distance into Minefield (yards)	1143	571	381	286	229	190	163	143	127	114
3000	Distance into Minefield (yards)	1333	667	444	333	267	222	190	167	148	133
2500	Distance into Minefield (yards)	1600	800	533	400	320	267	229	200	178	160
2000	Distance into Minefield (yards)	2000	1000	667	500	400	333	286	250	222	200
1500	Distance into Minefield (yards)	2667	1333	889	667	533	444	381	333	296	267
1000	Distance into Minefield (yards)	4000	2000	1333	1000	800	667	571	500	444	400
500	Distance into Minefield (yards)	8000	4000	2667	2000	1600	1333	1143	1000	889	800

Table 4. “First-Attempt” Evaluative TDA for Distance Into Minefield.

An example will show what is wrong with this TDA. Assume a submarine has intelligence reports that the enemy is laying mines at a rate of five mines per square nautical mile. The sub’s sensor has a known sweep width of 2000 yards in the given acoustic environment. After encountering a mine, the submarine CO might decide to back up at least 400 yards using this table (as shown circled in Table 4). The answer of 400 yards is merely the expected value of an exponential distribution. What is the probability that the actual distance is greater than 400 yards? For an exponential distribution, $P(X > x) = 1 - F(x;\lambda) = 1-(1-e^{-\lambda x})$. Therefore,

$$P(X > E(X)) = 1 - F(E(X);\lambda) = 1-(1-e^{-\lambda E(X)}) = 1-(1-e^{-\delta w \frac{1}{\delta w}}) = e^{-1} = 0.63.$$

In other words, if the submarine backed up 400 yards, 37% of the time this would not be far enough. Rather than using E(X) as a basis for backing up, what other values would be better?

Distance into Minefield	Probability that Distance is Far Enough
E(X)	0.63
2*E(X)	0.86
3*E(X)	0.95
4*E(X)	0.98
5*E(X)	0.99

Table 5. Determining a Useful Estimate for How Far to Backup.

From Table 5 above, a value of 3*E(X) is shown to provide a good recommended distance 95% of the time. Due to the exponential distribution, traveling further only makes marginal improvements. Therefore, the value of 3*E(X) is the basis for the tactical decision aid provided in the next section.

D. TACTICAL DECISION AID

With the original example given in section C (δ=2.5 mines/nm^2, w=1 nm), the submarine would now choose to back up 1200 yards after detecting the first mine. Table 6 illustrates this below.

		Minefield Density (mines per square nm)									
Sweep Width (yards)		1	2	3	4	5	6	7	8	9	10
4000	Distance into Minefield (yards)	3000	1500	1000	750	600	500	429	375	333	300
3500	Distance into Minefield (yards)	3429	1714	1143	857	686	571	490	429	381	343
3000	Distance into Minefield (yards)	4000	2000	1333	1000	800	667	571	500	444	400
2500	Distance into Minefield (yards)	4800	2400	1600	1200	960	800	686	600	533	480
2000	Distance into Minefield (yards)	6000	3000	2000	1500	1200	1000	857	750	667	600
1500	Distance into Minefield (yards)	8000	4000	2667	2000	1600	1333	1143	1000	889	800
1000	Distance into Minefield (yards)	12000	6000	4000	3000	2400	2000	1714	1500	1333	1200
500	Distance into Minefield (yards)	24000	12000	8000	6000	4800	4000	3429	3000	2667	2400

Table 6. Evaluative TDA for Distance Into Minefield.

Of course, the "expected minefield density" is the difficult input in using this TDA. For example, what if the CO does not trust the intelligence report and believes the minefield is as sparse as one mine per nm, he may decide to back up 6000 yards (as shown circled in Table 7). If the CO trusts the density assumption but the sonar conditions are unreliable (therefore making sweep width less accurate), the CO could back up anywhere from 600 to 4800 yards (for range of reasonable sweep widths) as shown shaded in Table 7.

		Minefield Density (mines per square nm)									
Sweep Width (yards)		1	2	3	4	5	6	7	8	9	10
4000	Distance into Minefield (yards)	3000	1500	1000	750	600	500	429	375	333	300
3500	Distance into Minefield (yards)	3429	1714	1143	857	686	571	490	429	381	343
3000	Distance into Minefield (yards)	4000	2000	1333	1000	800	667	571	500	444	400
2500	Distance into Minefield (yards)	4800	2400	1600	1200	960	800	686	600	533	480
2000	Distance into Minefield (yards)	6000	3000	2000	1500	1200	1000	857	750	667	600
1500	Distance into Minefield (yards)	8000	4000	2667	2000	1600	1333	1143	1000	889	800
1000	Distance into Minefield (yards)	12000	6000	4000	3000	2400	2000	1714	1500	1333	1200
500	Distance into Minefield (yards)	24000	12000	8000	6000	4800	4000	3429	3000	2667	2400

Table 7. TDA Showing Range of Options Depending on Reliability of Input Criteria.

E. CAUTION

The TDA for distance into minefield relies heavily upon some significant assumptions. The key assumptions that warrant caution are 1) Poisson distribution of mines 2) knowing the expected minefield density 3) perfect detection within sweep width and 4) rectangular minefield perpendicular to track.

Although minelayers typically steer along straight lines when laying mines, the cumulative effect of navigation errors, multiple lines, uneven spacing, current, and other errors results in a minefield in practice whose mines appear distributed roughly uniformly. Chapter 10 of Naval Operations Analysis provides more details of this argument. The decision maker can adjust for the next two assumptions (known minefield density and sweep width) as described in the previous section. The decision maker with uncertainty in these values can use the tables to ascertain a range of possible decision values. Finally, the model implicitly assumes the minefield is rectangular and laid perpendicular to the intended track. If after repeated mine detections, a submarine commander realizes a trend in the mine locations, he may be able to guess at the minefield orientation. Figure 13 shows this concept.

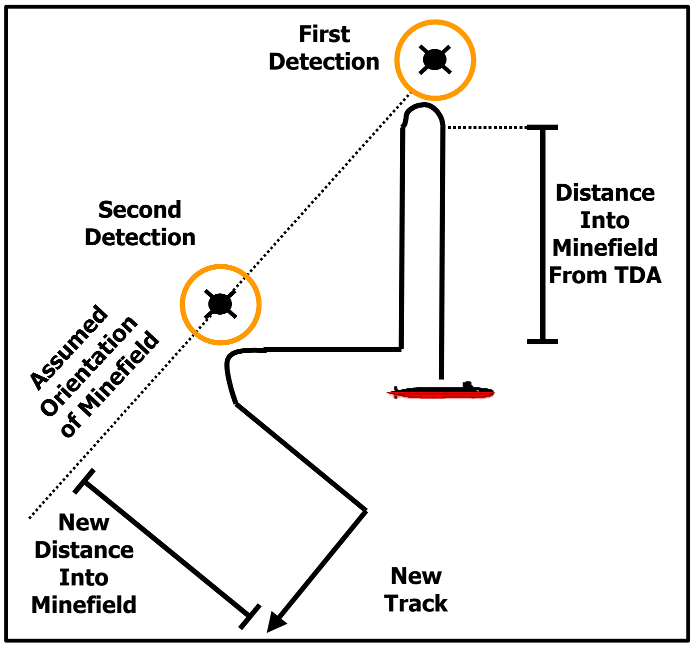

First Detection
Second Detection
Assumed Orientation of Minefield
Distance Into Minefield From TDA
New Distance Into Minefield
New Track

Figure 13. Distance into Minefield TDA with a Second Observation.

IV. MINIMUM STANDOFF DISTANCE

Mines generally make no noise as they wait for a target…this quality of mines strips submarines of their most effective defense – namely, the ability to hear danger before encountering it.

"Mine Warfare and Submarines," Jim Crimmins, Proceedings

A. PROBLEM DEFINITION

Two scenarios would lead a submarine to avoid a mine contact. Either the submarine has detected the mine with its onboard active sonar, or it is avoiding a reported mine position (detected by other assets or detected earlier by the submarine). In either case, the submarine must establish a minimum standoff distance at which it must order a maneuver to avoid the contact. Each possibility must take into account different factors.

The choice of minimum standoff distance is fundamental when modeling a submarine penetrating a minefield. Calculating this distance first provides a reasonable input to the simulation (described in the next chapter). Johns Hopkins Applied Physics Laboratory used similar calculations in the Advanced Vehicle Concept study. However, that study dealt mainly with an Unmanned Underwater Vehicle (UUV) and had different assumptions than those considered in this thesis.

B. MODEL DESCRIPTION

1. Calculating Lethal Range

Shock factor (SF) is a simple model of a mine's lethal range. As illustrated in Figure 14, shock factor depends on range, the TNT equivalent of the mine warhead, and the depression angle from the submarine to the mine.[12] For simplification, the model only considers shock factors above a certain threshold as kills. The model considers a submarine suffering multiple shock factors under this threshold as still having made a safe transit.

[12] Boerman, Douglas A. "Finding an Optimal Path Through a Mapped Minefield," (M.S. Thesis, Naval Postgraduate School, Monterey, California, March 1994): 19.

$$SF = \frac{\sqrt{W}}{R}\frac{1+\sin\alpha}{2}$$ where W = explosive's TNT equivalent weight (pounds)

R = Slant range from hull to mine (feet)

α = Depression angle between hull and mine (degrees)

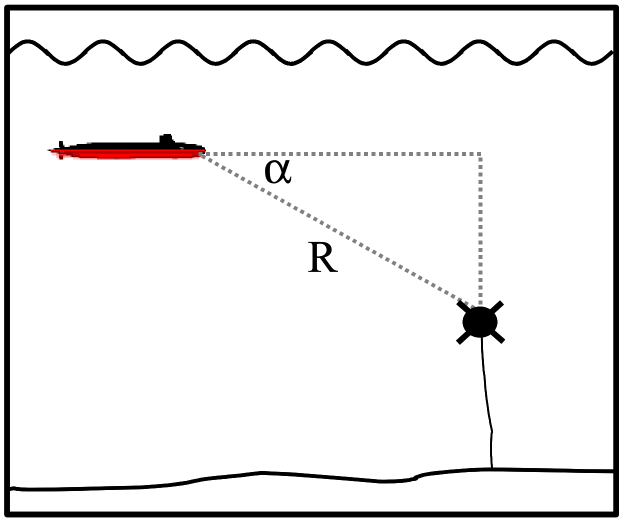


Figure 14. Shock Factor (After: Boerman, 1994, 20)

What shock factor is appropriate for determining a submarine’s minimum standoff distance from a mine? First, assume the worst-case depression angle (α) of 90 degrees (the mine detonates directly beneath the vessel). Second, assume a submarine’s lethal threshold shock factor is 0.5. An assumption of a 2000-pound TNT equivalent warhead results in a lethal range of 90 feet or 30 yards.

2. Avoiding a Mine Detected by Onboard Sonar

Figure 15 shows the case where a submarine detects the mine with active sonar. An example will use specific numbers and then follow up with a general equation. The advance and transfer are of equal values. The model assumes that the mine contact is directly on the submarine’s original course. If the contact is to the left of track, a turn to the right will only increase the margin of safety. Further, the submarine would not turn towards the contact. For example, if the contact were on the left, the sub would not try to turn to the left (even though some tactical situation may dictate such a maneuver). In addition, the actual sensor error is assumed to be a bivariate normal with a known standard deviation. The model assumes that the submarine has a radius of a “3 sigma” ellipse, encompassing the actual target 98.9% of the time.

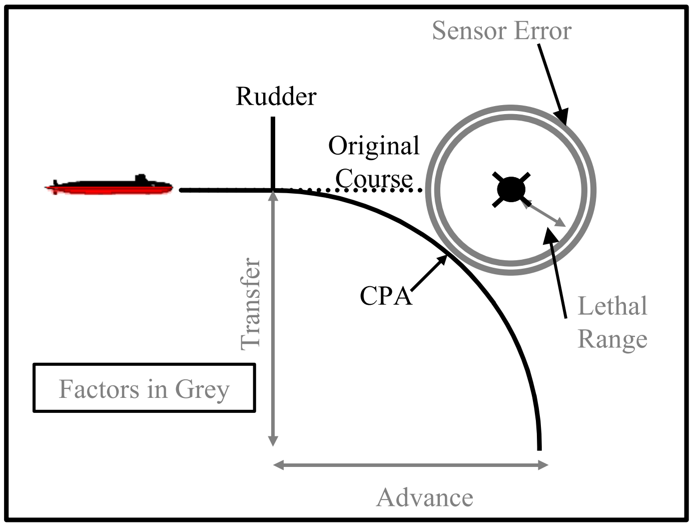
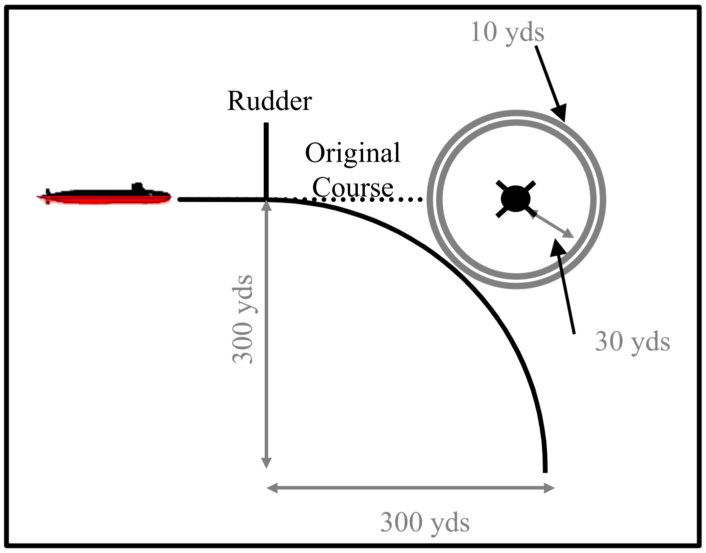


Figure 15. Minimum Standoff Distance Assuming Detection with Onboard Sonar (Not Drawn To Scale).

The advance and transfer are approximately 300 yards for a five-knot transit. The section on shock factor determined that a large warhead mine has a lethal range of 30 yards. The assumed sensor error is 10 yards and is consistent with that chosen by the Advanced Vehicle Concept study.[13] Again, the model assumes this value of 10 yards is the maximum radius of a "3 sigma" ellipse of the bivariate normal error term. Figure 16 shows all of these values.



[13] Johns Hopkins Applied Physics Laboratory. Mission Requirements and Requirements Analysis for the Advanced Vehicle Concept. JWR-97-017. (Laurel, MD 1992), 4-1.

Figure 16. Nominal Values Assuming Detection with Onboard Sonar (Not Drawn To Scale).

3. Avoiding a Previously Reported Mine Position

In the second case, the advance, transfer, and lethal ranges have the same values as in the first example. However, the mine's position error is now due to uncertainty of information. Specifically, this error includes: 1) sensor error of platform that detected contact (e.g. UUV) 2) plotting error (mine position is plotted on submarine's navigation chart) 3) uncertainty over time (a contact report 10 days old will not be well trusted). Just as in determining the sensor error in the last case, the model assumes this value is the maximum radius of a "3 sigma" ellipse of the bivariate normal error term. The example uses a mine position error of 50 yards. The circular navigation error includes unaccounted set and drift errors as well as inertial position error. The example assumes a navigation error of 200 yards. Therefore, the standard deviation of this bivariate normal position error is 66.6 yards. Navigation error does not apply during active sonar detection since the relative position of the submarine and mine is known within the accuracy of the sensor. The following two figures show the factors and the nominal values. Note that at Closest Point of Approach (CPA), the navigation error circle is tangent to the mine position error circle.

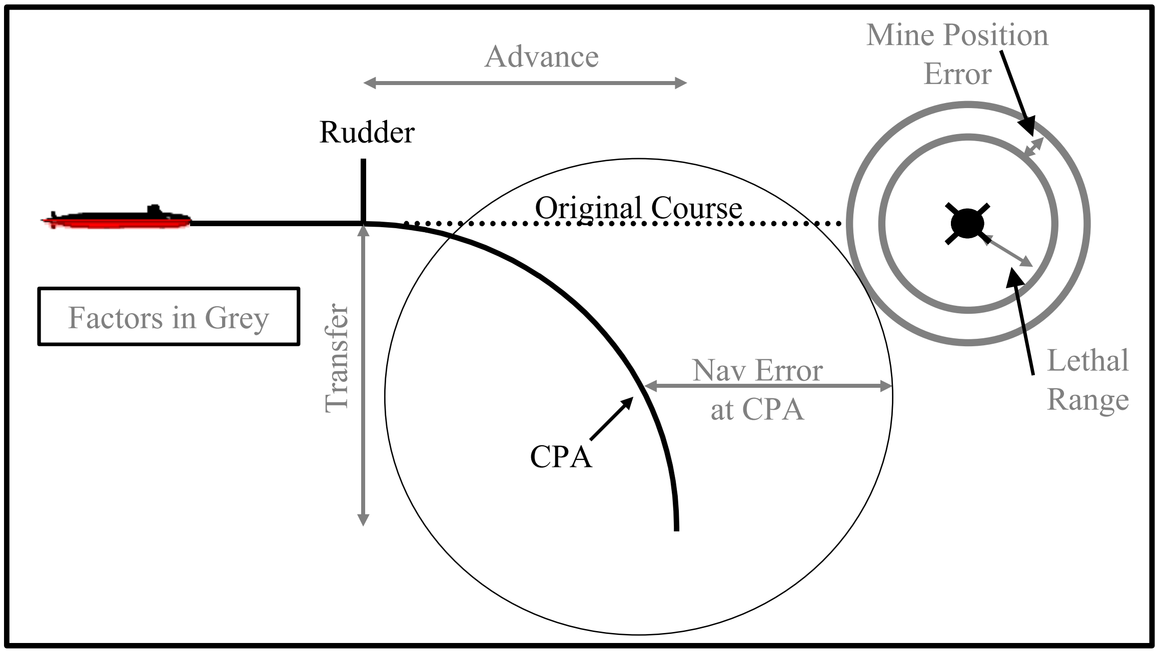


Figure 17. Minimum Standoff Distance Assuming Avoidance of a Previously Reported Mine Position (Not Drawn To Scale).

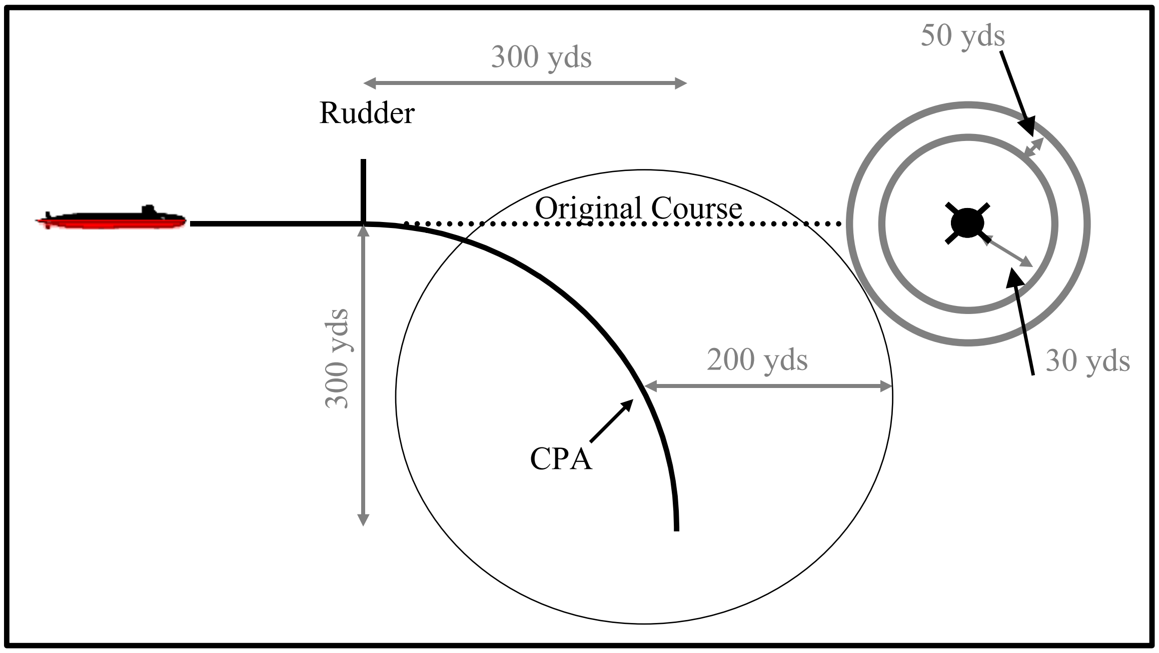


Figure 18. Nominal Values Assuming Avoidance of a Previously Reported Mine Position (Not Drawn To Scale).

C. ANALYSIS

From these scenarios, the distance from rudder order to mine position is shown below.

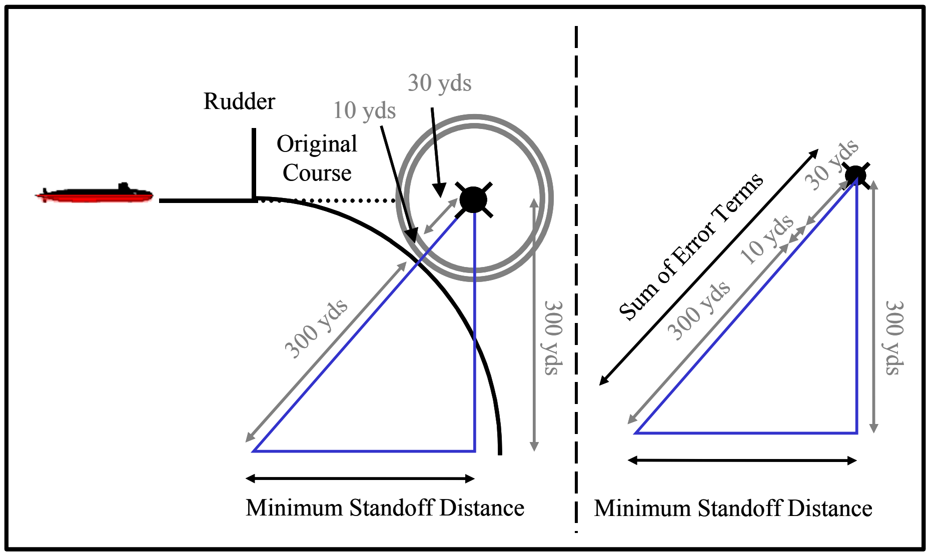


Figure 19. Calculating Minimum Standoff Distance Assuming Detection with Onboard Sensor (Not Drawn To Scale).

Therefore, in the case where the submarine detects the mine,

$$\text{Minimum Standoff Distance} = \sqrt{(300+10+30)^2 - (300)^2} = 160.0 \text{ yds}.$$

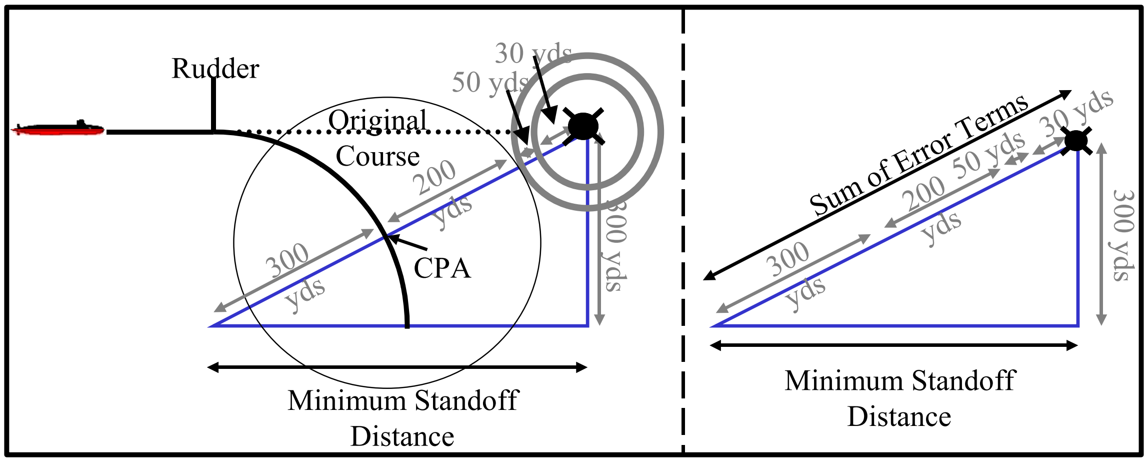


Figure 20. Calculating Minimum Standoff Distance Assuming Avoidance of a Previously Reported Mine Position (Not Drawn To Scale).

In the second case where the submarine is avoiding a mine position without regaining detection, both the navigation error and mine position error are from “3 sigma” error circles. Therefore, when summing those two error terms ($\text{error}_{\text{Sub}}$ and $\text{error}_{\text{Mine}}$) the total error from those two effects is $\sqrt{\text{error}_{\text{Sub}}^2 + \text{error}_{\text{Mine}}^2}$. Therefore, in the case where the submarine avoids a previously reported mine position:

$$\text{Minimum Standoff Distance} = \sqrt{\left(300+\sqrt{200^2+50^2}+30\right)^2 - (300)^2} = 444.4 \text{ yds}$$

D. RESULTS

1. Minimum Standoff Distance When Detecting With Onboard Sonar

For the example in the previous section, a submarine that detects a contact ahead must commence a turn at least 160.0 yards away to ensure the submarine stays clear with the given assumed values. In general, the minimum standoff distance (yards) for a submarine after detecting a mine is shown in Figure 21.

$$\boxed{\begin{aligned}&\text{Minimum Standoff Distance (yds)}\\ &\quad = \sqrt{(\text{advance} + \text{sensor error} + \text{lethal range})^2 + (\text{transfer})^2}\end{aligned}}$$

Figure 21. General Equation for Minimum Standoff Distance Assuming Detection.

2. Minimum Standoff Distance When Avoiding a Previously Reported Mine Position

For the second case where the submarine does not have an active sonar detection of the mine, the submarine must turn at least 496.4 yards before the assumed mine position. In general, the minimum standoff distance (yards) for a submarine avoiding a previously reported mine position is shown in Figure 21.

$$\boxed{\begin{array}{l}\text{Minimum Standoff Distance (yds)} = \\ \sqrt{\left(\text{advance} + \sqrt{\text{nav error}^2 + \text{mine position error}^2} + \text{lethal range}\right)^2 + (\text{transfer})^2}\end{array}}$$

Figure 22. General Equation for Minimum Standoff Distance Assuming Avoidance of a Previously Reported Mine Position.

The simulation in Chapters V and VI uses the value of 160.0 yards and 450.0 yards (rounded from 444.4) for each of the two cases.

3. Impact on Sonar Range Performance

As a submarine approaches a mine, the mine becomes detectable when within the range of the submarine's sonar. After the mine is within detectable range, there is some delay until the sonar operator or Computer Aided Detection (CAD) detects the contact. Then, the sonar operator must decide on whether or not the contact is a mine, and the Officer of the Deck (OOD) decides whether the submarine will maneuver. Finally, if necessary, the submarine maneuvers around the assumed mine.

A single delay term can group the delay between initial detection and contact decision and the subsequent delay for a rudder order. The Advanced Vehicle Concept assumed a delay of roughly 30 seconds for a UUV. For example, this model assigns a delay term a value of 1.2 minutes based upon a manned submarine. This equates to 200 yards when traveling at 5 knots. Therefore, with a delay of up to 200 yards, the submarine needs sonar capable of detecting contacts at 160.0+200 = 360.0 yards range.

THIS PAGE INTENTIONALLY LEFT BLANK

V. PROBABILITY OF SAFE TRANSIT

Our work indicates that effective mine avoidance and minefield penetration by a submarine with unmanned vehicles are not impossible. However, they are difficult.

VADM Grossenbacher, COMSUBLANT[14]

A. PROBLEM DEFINITION

The principal measure of effectiveness for a submarine conducting a minefield penetration is the probability of safe transit (P_{st}). With various mine densities and NOMBO densities, what is the chance that the submarine will safely transit across the minefield? The P_{st} depends upon the environment, the mines, and the submarine. Assumptions about these three factors are essential in developing a model.

B. MODELING ASSUMPTIONS

The following subsections describe the range of assumptions about these three factors. Then, the particular assumptions kept for this model are explained.

1. Environment

The level of detail in the modeling of the environment affects both the mine and submarine models. Assumptions about the environment affect the detection model of the submarine and the mine. It also affects the actuation model for the mine. The set and drift caused by current affect the submarine's navigation error. Whether or not to model the environment as two-dimensional or three-dimensional is a fundamental assumption. The chosen model of water depth affects the type of mines that can be considered (moored mines can be placed in water much deeper than bottom mines).

The model developed in this thesis considers the environment only in two dimensions. The model implicitly assumes a shallow water chokepoint transit with depths less than 300 feet since both moored and bottom mines are included. The effects of the environment upon detection models is broadly generalized since this model uses cookie cutter sensors with an assigned probability of detection at maximum range. The probabilities and ranges of detection are appropriate for a shallow water chokepoint

[14] Grossenbacher, John J. "Remarks at 2002 NDIA Clambake," The Submarine Review, (January 2003), 12.

transit. Finally, bottom-clutter (NOMBOs) occurs randomly according to a Spatial Poisson process (as explained in Appendix B).

2. Mines

Modeling mines includes assumptions about the detection, actuation, mine type, and mine dispersion. First, mines have acoustic, magnetic, pressure, and other sensors. All of these have complex underlying models that can affect detection and actuation. Secondly, different mine types can be modeled (moored mines, bottom mines, drifting mines, etc…). The type of mine dispersion (in lines, randomly in fields, systematically, etc…) is another important modeling assumption. In addition, a model must account for the mines lethal range. One method is to simplify with threshold values for shock factor as described in Chapter IV. It is important to realize that there are over 300 mine types in the world. Many of these have unknown capabilities or have microprocessor components that can be readily modified, which results in imperfect knowledge of true mine performance. Accordingly, this model uses some generic simplifications.

The model has both moored and bottom mines with different values for maximum detection range. The simulation distributes mines in a field according to a spatial Poisson process (explained in Appendix B). The simulation keeps a number of assumptions about mines from the CNA study:[15]

- Mines are 100% reliable
- Cookie-cutter sensor for mine
- No sonar error for range or bearing
- Ignore case tilt for mines
- Two dimensional model
- No false contacts

3. Submarine

Modeling a submarine includes assumptions about the navigation, maneuvering, and detection models. Since submarines use inertial navigation, error is inherent with the

[15] Schaffer, Matt. "Follow-On Analysis of Minefield Avoidance and Penetration." (Center for Naval Analyses, CRM 91-40), 1991.

submarine's own position. The model can incorporate or ignore this error in the model. Another factor in navigation error occurs if the environmental model includes set and drift. Unaccounted set and drift error can be an additional error source for navigation. A submarine's motion model depends on whether it is two-dimensional or three-dimensional. The most complex maneuvering model is the six-degree-of-freedom motion model. Simplifying models include use of turn radius or advance and transfer. Models could also include acceleration as another aspect of maneuvering. The possibility of false alarms is another modeling issue. Finally, the detection model can be anything from a complex acoustic model to a simple cookie cutter sensor.

The final model used in this simulation includes a submarine with a cookie cutter sensor (different ranges for different mine types) maneuvering across a barrier of moored mines, bottom mines, and NOMBOs. The submarine has a navigation error due to inertial navigation errors and unaccounted set and drift. The submarine moves at a constant speed and turns are based upon advance and transfer at the given speed. Just as in the CNA study, a simulated submarine will only change its course if it 1) needs to avoid a new contact 2) needs to avoid an old contact 3) needs to adjust its course (if not in the general direction of the destination).

Similar studies of MDA have not included the effects of a submarine's navigation errors. This can be a shortcoming, since inertial navigation system errors are especially significant when in a minefield. The errors of inertial navigation can increase under slow speeds and during frequent drastic maneuver – just the type of maneuvering expected in MDA operations. If the active sonar has little position error (assumed to be zero error in this study) upon detecting a mine, why does navigation error matter? During MDA, a submarine may have to maneuver based upon the plotted position of a previously reported mine. As illustrated in Figure 23, the submarine may not detect the original contact again. Because of navigation errors, the submarine's relative position may place its track right over the original mine contact rather than avoiding it. With navigation errors being in the range of hundreds of yards, models must account for this distinct possibility. This simulation takes into account navigation error implicitly by using the minimum standoff distance of 160 yards vice the 450-yard standoff distance discussed in Chapter IV.

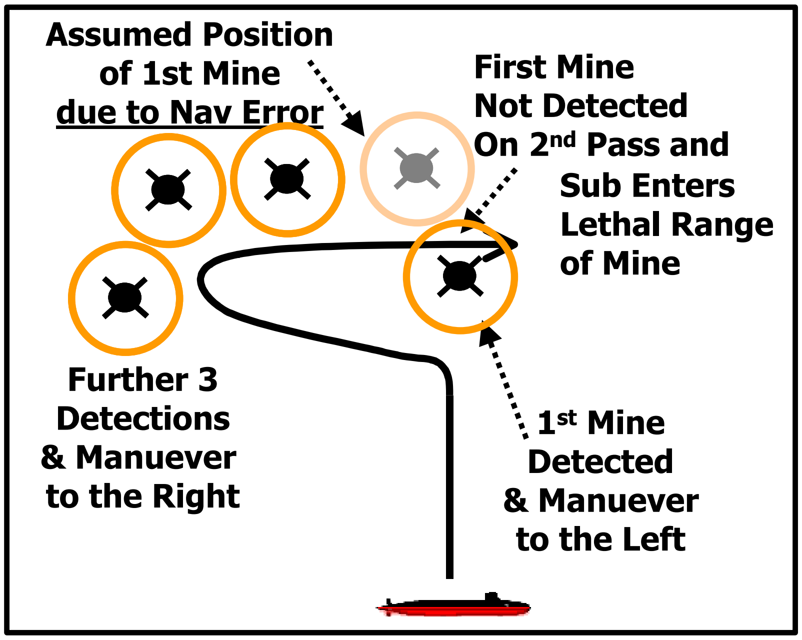


Figure 23. Navigation Error Resulting in Mine Actuation.

C. MODEL DEVELOPMENT

The simulation approach was compared to previous analytical methods and other simulation results. The analytical solutions did not include the submarine's maneuvering characteristics.

1. Discrete-Event Simulation

Discrete-event simulation considers a set of events based upon state variables and parameters of simulation entities. For example, a mine could have the parameters of location and sensor range. The beginning of every replication fixes the parameters. The mine's state variable would be a Boolean called "actuated." Initially the actuated variable would be set to false. When the submarine began to move in the model, the mine's actuated variable would change to true once the event "submarine enters within sensor range of mine" occurred. For more details about discrete-event simulation modeling, refer to Law and Kelton's text, Simulation Modeling and Analysis.

2. Event Graphs and Simkit

Event graphs are "a way of graphically representing discrete-event simulation models."[16] Lee W. Scruben first developed event graphs in 1983. Their simple

[16] Arnold Buss. "Basic Event Graph Modeling," Simulation News Europe (April 2001): 1.

construction and ability to encompass all aspects of a discrete event model make them useful. Simkit is a package written in Java for developing Discrete-Event Simulation models that directly supports event graphs since, "For every element in an EG [Event Graph] there is specific Java code that Simkit interprets"[17] (see Table 8).

EG	Simkit
State Variables	Class instance variable
Simulation Parameters	Class instance variable
Events	Class "do" method
Event Parameters	Parameter of a "do" method
Event Actions	Code lines of a "do" method
Scheduling Edges	"waitDelay" call
Canceling edges	"interrupt" call
Edge Delay Times	Time argument of "waitDelay" call
Edge Conditions	"if condition block", wrapping a "waitDelay" or "interrupt" call.
Edge Arguments.	"waitDelay" or "interrupt" call arguments

Table 8. Relationship between Event Graphs and Simkit (From: San Jose, 2001, 3).

Simkit also consists of building blocks of code very transferable to the minefield problem. Simkit has "mover" objects that it uses to represent the mines, the NOMBOs, and the transiting submarine. Each mover can have an associated "sensor" class. Simkit's "referee" class knows the location and velocity of all movers and notifies a "mediator" when one mover is to enter or exit the sensor range of another mover. The mediator class then uses a rule set (such as information on P_d) to determine when the sensor will actually detect the target. Another class the "path mover manager" describes how each mover object makes decisions on moving from one point to another. This class

[17] Angel San Jose, "Analysis, Design, Implementation and Evaluation of Graphical Design Tool to Develop Discrete Event Simulation Models Using Event Graphs and Simkit." M.S. Thesis, Naval Postgraduate School, 2.

gave the submarine a path planning capability. Building a simulation with this method easily allowed for upgrades.

3. Common Random Number Streams

Each replication of the simulation draws a new number of mines according to a Poisson distribution. The simulation placed these mines such that their x and y positions were generated from independent uniform distributions. At the beginning of each run, the simulation reset the random number seed generating the minefield. This is a simulation practice known as setting common random number streams. Therefore, any effect on the measured output (probability of safe transit) could be traced to a change in parameters, not to different random minefields.[18] For example, when comparing probability of detection of 0.5 and 0.8, replication i for the simulation run with $P_d = 0.5$ was the same as replication i for the simulation run with $P_d = 0.8$.

D. SIMPLE MINEFIELD TRANSIT MODEL

The Simkit simulation results were compared to those from Chihoon Kim's thesis. Just as in the results of Kim's work, more complex maneuvering models showed more optimistic results. As suggested in Kim's thesis, this model explored a more complex avoidance tactic.

1. Comparison to Analytical Model

As a baseline comparison, the simulation incorporated Kim's Simple Minefield Transit (SMT) obstacle avoidance algorithm. In this model, if a submarine detects a mine, it backs up to the beginning of the minefield, picks a new entry point and tries to cross again. Essentially, the submarine is trying to find a straight-line transit without any mines in its path. Figure 24 illustrates the SMT model.

[18] Sanchez, Susan. "ABC's of Output Analysis." (Proceedings of the 2001 Winter Simulation Conference, B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, eds., 2000): 36.

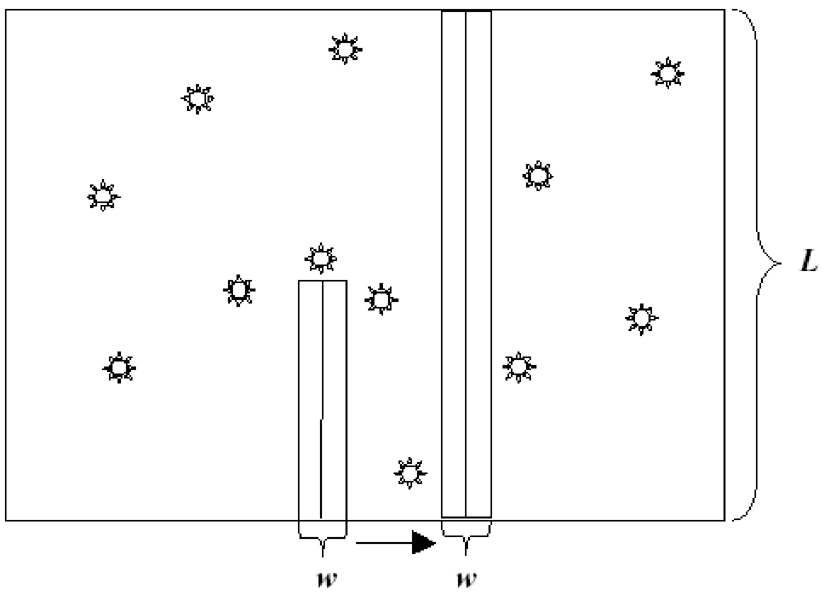


Figure 24. SMT Model of Minefield Transit (From: Kim, 2002, 11).

As discussed in Chapter II, Kim developed an analytical equation for this SMT model. This served as a baseline comparison for Kim's simulation. In the mine only case (no NOMBOs or false contacts), the equation reduced to:[19]

$$P_{st} = \frac{e^{-\lambda_M L w}}{1-(1-e^{-\lambda_M L w})P_d}$$

Kim's simulation replicated the SMT model 10,000 times with the following parameters:

L (minefield length) = 6 nm

w (mine actuation diameter) = 1 nm

Therefore, actuation range = 1000 yds

λ_M (rate of mines) = 0.3 mines/nm^2

P_d (probability submarine detects mines) = variable

The only additional parameter was λ_O (rate of NOMBOs) = 0.3 mines/nm^2, during the "mine and NOMBO" case. When implementing this simulation with Simkit, the minefield width was set at 100 nm. Both the analytical model and the SMT simulation assumed an infinitely wide minefield, so the submarine never ran out of reentry points.

[19] Chihoon Kim, "The Effect of Sensor Performance on Safe Minefield Transit" M.S. Thesis, Naval Postgraduate School: 14.

With an infinite width minefield, the submarine transit had two end states: safe transit or actuate a mine. With a finite width minefield, the submarine has an additional end state: "gridlock" i.e., the probability that a single attempt to penetrate the minefield is blocked by mines or NOMBOs. *For the rest of this chapter, the P_{st} computed is conditioned on the probability that the transit did not result in gridlock.* This allows comparisons to the analytical model. Chapter VI discusses the probability of gridlock in more depth.

In addition, the Simkit model gave the submarine a cookie cutter sensor with an associated probability of detection. The simulation used this "probability cookie cutter" with a range of 1000 yards (the same as the mine's actuation range). The detection range of the SMT model was implicitly the same as the mine actuation range. The SMT model was independent of speed (only distance traveled was measured). The Simkit model needs an arbitrary speed (chosen to be 5 knots) to schedule times for events. Figure 25 displays the event graph for the SMT model.

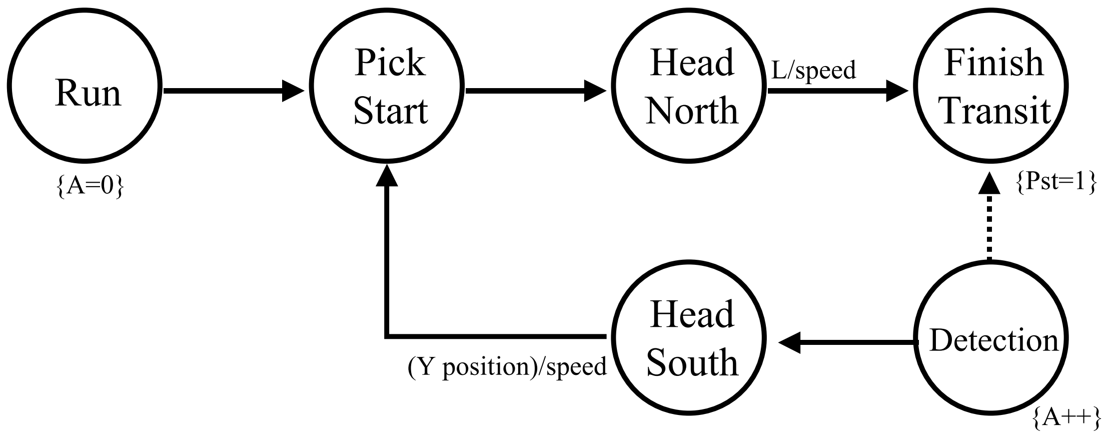


Figure 25. Event Graph for SMT Model.

In this event graph, the parameter A keeps track of the number of detections. The first event in an event graph is always "Run." Run immediately schedules a "Pick Start" event. The submarine starts 1000 yards (w/2) to the right of the far left side of the minefield. The submarine then conducts a "Head North" event. The "Head North" event schedules a "Finish Transit" to occur at time L/speed where L is the minefield length. Any "Detection" event (based upon the submarines position and the random scattering of the mines) cancels the "Finish Transit" event and schedules a "Head South." Head South schedules a "Pick Start" only after the time it takes to get back to the front of the minefield (in this case, current Y position over speed). The new start point is incremented 2000 yards (w) over to the right from the last entry point. This choice is made clear in Figure 26.

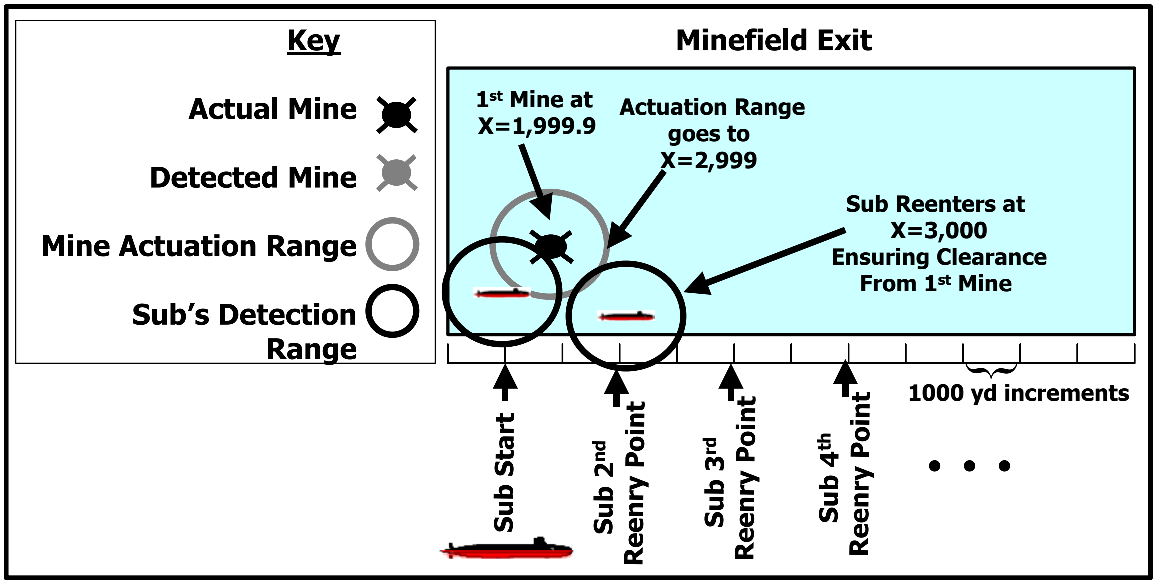


Figure 26. Incrementing Reentry Points for SMT Model.

2. Results of Comparison to SMT MODEL

Table 9 displays a comparison of the "mine only" case between the Simkit SMT model simulation and the analytical mean. The simulation's 95% confidence interval contains the analytical mean for all ten cases of probability of detection. The simulation stopped running once the standard error reached 0.015. Therefore, the number of runs varied as shown in Tables 9 and 10. Table 10 shows the "mine and NOMBO" case comparison. Again, the simulation's 95% confidence interval contains the analytical mean for all ten cases of probability of detection. Therefore, the simulation's results are consistent with the analytical model.

Pd		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Mine Only	+.95 CI	0.188	0.225	0.249	0.264	0.285	0.341	0.386	0.443	0.557	0.706	1.000
	-.95 CI	0.128	0.165	0.189	0.204	0.225	0.281	0.326	0.383	0.497	0.646	0.970
	Mean	**0.158**	**0.195**	**0.219**	**0.234**	**0.255**	**0.311**	**0.356**	**0.413**	**0.527**	**0.676**	**1.000**
	Std Error	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
	# of Runs	548	672	731	766	888	916	980	1036	1066	936	412
Analytical Mean		**0.165**	**0.180**	**0.198**	**0.221**	**0.248**	**0.284**	**0.331**	**0.398**	**0.498**	**0.664**	**1.000**

Table 9. Comparison of Simulation and Analytical Mean (SMT "Mine Only" Case).

Pd		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Mine & NOMBOs	+.95 CI	0.198	0.200	0.202	0.226	0.255	0.288	0.340	0.402	0.509	0.670	1.000
	-.95 CI	0.138	0.140	0.142	0.166	0.195	0.228	0.280	0.342	0.449	0.610	0.970
	Mean	**0.168**	**0.170**	**0.172**	**0.196**	**0.225**	**0.258**	**0.310**	**0.372**	**0.479**	**0.640**	**1.000**
	Std Error	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
	# of Runs	597	605	610	620	575	649	672	202	145	192	145
Analytical Mean		**0.165**	**0.180**	**0.198**	**0.221**	**0.248**	**0.284**	**0.331**	**0.398**	**0.498**	**0.664**	**1.000**

Table 10. Comparison of Simulation and Analytical Mean (SMT "Mine & NOMBO" Case).

E. LATERAL EXCURSION AVOIDANCE MANEUVER

1. Developing Lateral Excursion Avoidance Maneuver Model

Once a comparison showed the simulation to be consistent with the SMT model baseline, the next step was to explore more detailed obstacle avoidance algorithms. Kim had shown that his more detailed obstacle avoidance algorithm, termed "Minefield Object Avoidance Maneuver" (MOAM), had higher probabilities of safe transit. In this model, if the submarine detects a contact, it tries to go to the left some distance, $\overline{h}$. If the submarine does not detect another contact, it turns again and continues to the end of the field. If the submarine detects yet another contact, it would try to go the right (the same distance $\overline{h}$). If detecting new contacts at the left and right, the submarine would then back up along its original path and pick a new point to reenter the minefield. Figure 27 illustrates this below. Figure 28 displays the event graph for the MOAM model. The avoidance distance ($\overline{\text{h}}$) is set equal to mine actuation diameter (w) with the same logic as shown in Figure 26.

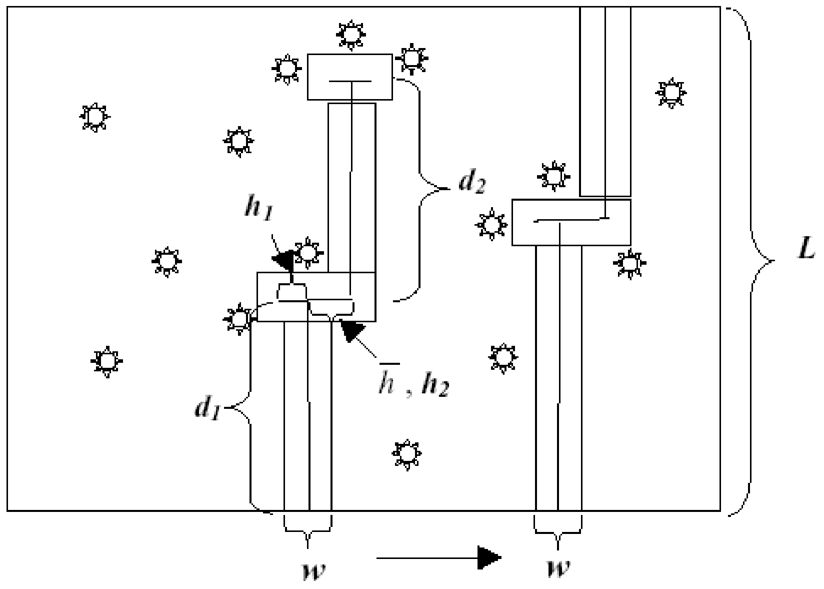


Figure 27. MOAM Model of Minefield Transit (From: Kim, 2002, 47).

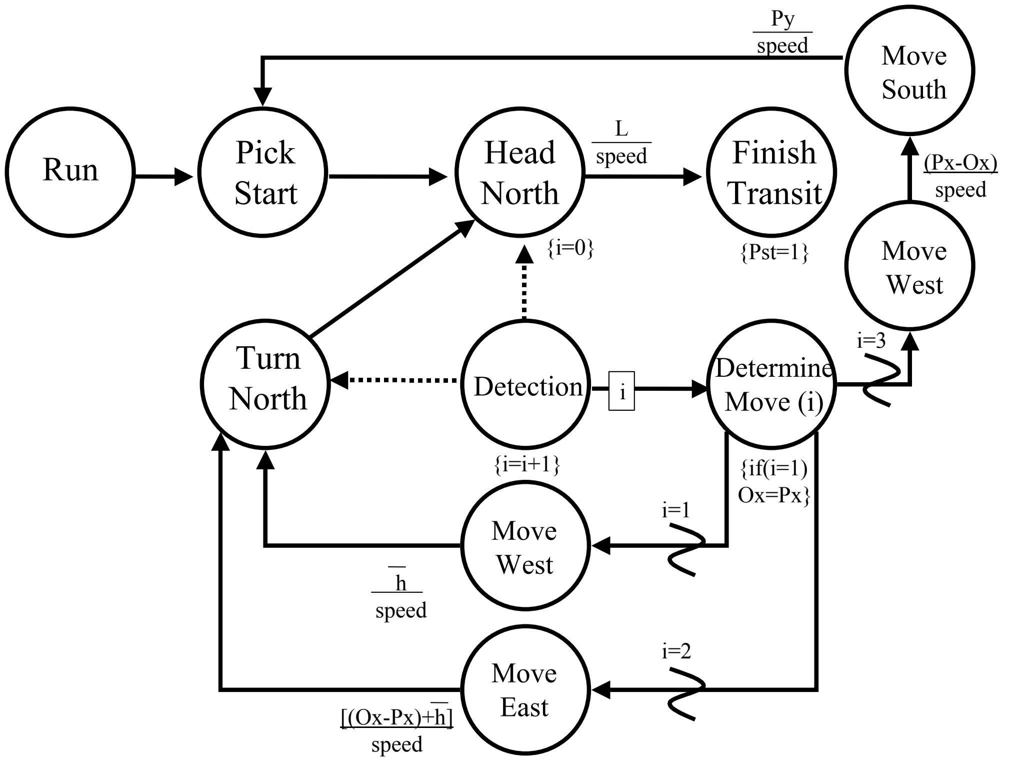


Figure 28. Event Graph for MOAM Model with Infinite Width Minefield.

Simkit does not directly implement the MOAM model. Kim's thesis suggested exploring other obstacle avoidance models. For example, using the MOAM model, if a submarine had transited across a 15 mile long minefield and encountered three objects blocking its path at the 14.5 mile point, the submarine would transit all the way back to the start and try again. An improved obstacle avoidance algorithm would try to get around these remaining obstacles rather than start over.

Under the new obstacle avoidance method, when a submarine detects a mine (or NOMBO) it maneuvers (either east or west) around the object and continues north. If the submarine detects more obstacles while avoiding the first, the submarine continues avoiding them in the same direction originally chosen. In other words, the submarine picks one direction for its lateral excursion and does not try to go back the other direction. This method is termed the Lateral Excursion Avoidance Maneuver (LEAM). The choice of direction must remain consistent, but can be either east or west. All following discussions will have the default direction chosen as east. The LEAM model is illustrated below in Figure 29.

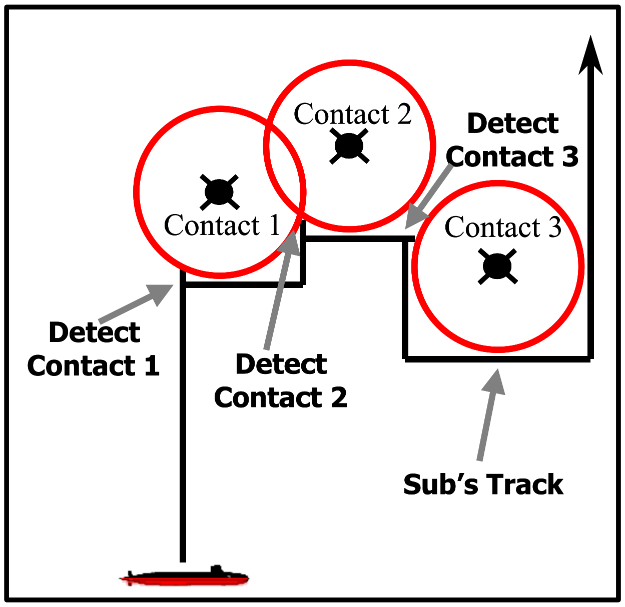


Figure 29. Lateral Excursion Avoidance Method.

One important difference between the LEAM and MOAM models is the choice of avoidance distance. The MOAM model sets this distance (termed $\bar{h}$) as twice the mine actuation range. It does not take into account information about the detected contact's position when conducting an avoidance maneuver. The LEAM model computes its avoidance distance as the contact's x-position plus actuation range (if avoiding to the right). Figure 30 highlights these differences.

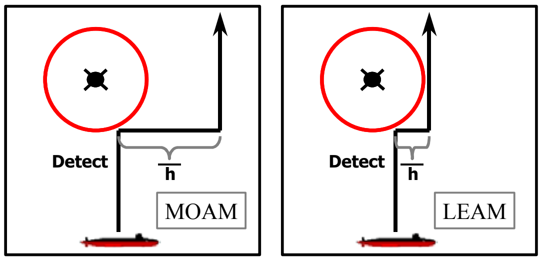


Figure 30. Differences in Models Choice of Avoidance Distance.

To construct the event graph of the LEAM model, it is easiest to build upon the event graph of the MOAM model. As noted, the MOAM model event graph developed depends upon the fact that the minefield has infinite width. In the initial Simkit implementation, the minefield is only 100 nm wide. Therefore, before attempting a lateral excursion, the algorithm must check whether the submarine will exceed the boundaries. In this case, before moving east, the algorithm checks that the current X position (P_x) plus $\bar{h}$ is greater than or equal to zero (left hand limit of minefield). Likewise, before moving East, the algorithm checks that $P_x+\bar{h}$ is less than or equal to w (minefield width). If either check fails (excursion is trying to go outside the boundaries of the minefield), then the submarine returns to the start and picks a new entry point. Simkit could implement the MOAM model with the event graph below in Figure 31.

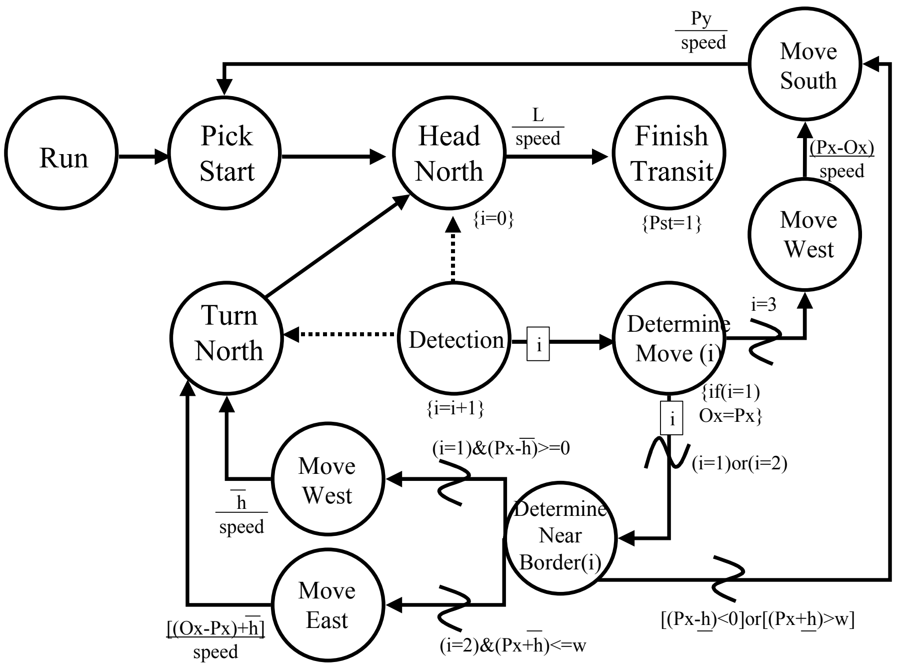


Figure 31. Event Graph for MOAM Model with Finite Width Minefield.

The LEAM model benefits from its consistent choice of lateral excursion direction. Before conducting an avoidance excursion to the east, LEAM must first check that the maneuver does not go past the edge of the minefield. Then, depending on whether the detected contact is north (simplified by comparing the contact's y-coordinate with the sub's y-coordinate) the algorithm chooses between two different avoidance maneuvers. Figure 32 displays the event graph for the baseline LEAM model.

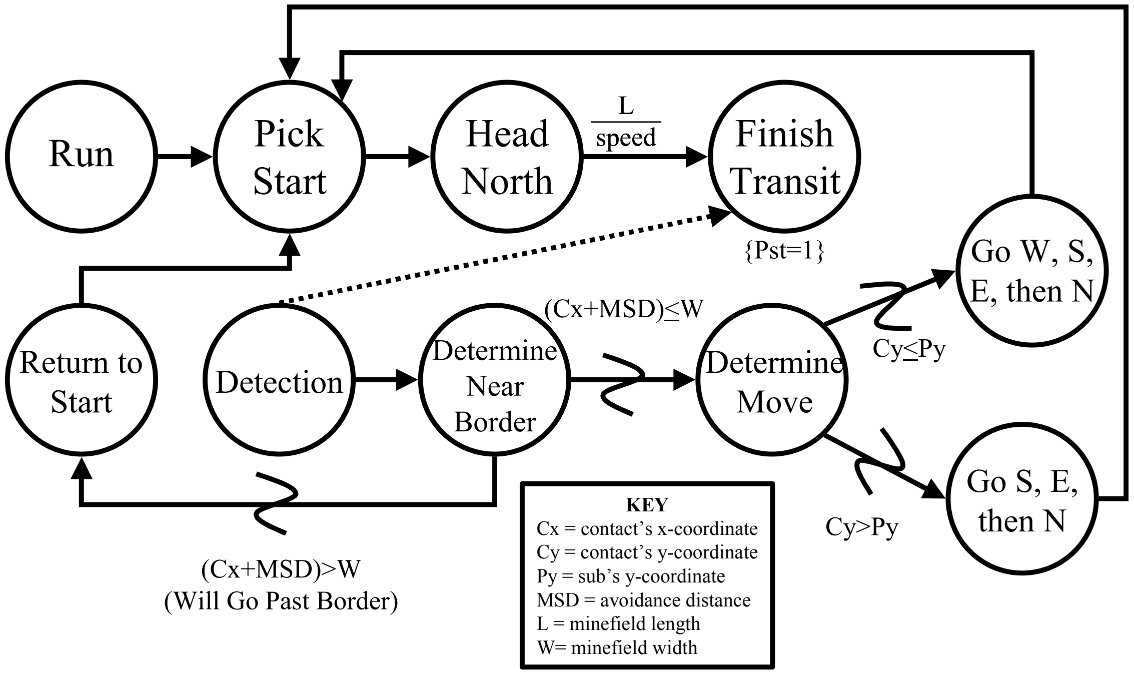


Figure 32. Event Graph for Baseline Lateral Excursion Avoidance Method.

2. Comparing LEAM to Other Models

As mentioned, earlier, the MOAM model has higher P_{st} results than the SMT model. The LEAM model should likewise have higher P_{st} results than the MOAM model. There is one other model that is of use in comparing the particular obstacle avoidance algorithm. The optimistic case discussed in Kim's thesis has the same assumptions as the SMT analytical model, except that upon detecting an object (mine or NOMBO), "it will proceed towards the end of the field without diversion and without exploding the mine."[20] This optimistic case serves as an upper bound for an obstacle avoidance algorithm. The equation for this upper bound is: $P_{st} = e^{-\lambda_M (Lw)(1-P_d)}$.

[20] Chihoon Kim, "The Effect of Sensor Performance on Safe Minefield Transit" M.S. Thesis, Naval Postgraduate School: 20.

Pd		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
LEAM Model Mine Only	+.95 CI	0.207	0.224	0.275	0.299	0.359	0.412	0.499	0.585	0.692	0.848	1.000
	-.95 CI	0.147	0.164	0.215	0.239	0.299	0.352	0.439	0.525	0.632	0.788	0.970
	Mean	**0.177**	**0.194**	**0.245**	**0.269**	**0.329**	**0.382**	**0.469**	**0.555**	**0.662**	**0.818**	**1.000**
	Std Error	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
	# of Runs	623	670	792	841	944	1009	1064	1056	956	637	128
SMT Analytical Mean		**0.165**	**0.18**	**0.198**	**0.221**	**0.248**	**0.284**	**0.331**	**0.398**	**0.498**	**0.664**	**1.000**
MOAM Analytical Mean		**0.165**	**0.190**	**0.219**	**0.255**	**0.299**	**0.354**	**0.424**	**0.513**	**0.630**	**0.785**	**1.000**
Analytical Upper Bound		**0.165**	**0.198**	**0.237**	**0.284**	**0.34**	**0.407**	**0.487**	**0.583**	**0.698**	**0.835**	**1.000**

Table 11. Comparison of LEAM Simulation and SMT, MOAM, and Upper Bound ("Mine Only" Case).

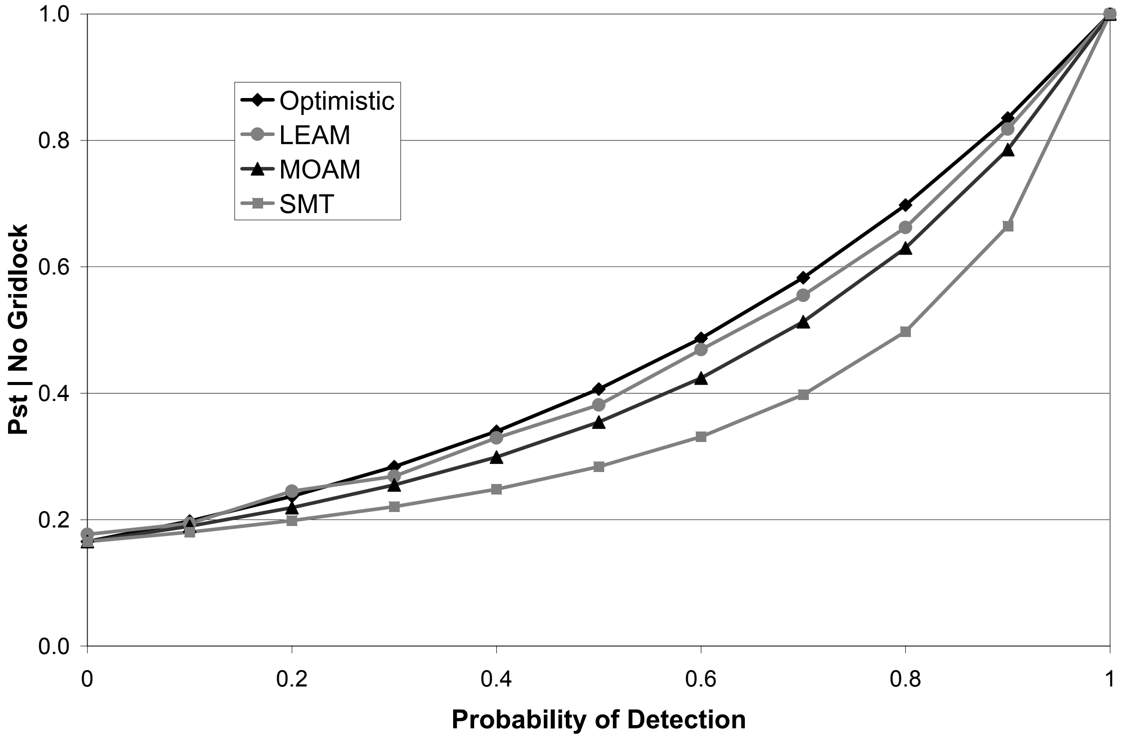


Figure 33. Comparison of Different Obstacle Avoidance Models ("Mine Only" Case).

As can be seen in Table 11 and Figure 33, the LEAM model achieves a P_{st} between the MOAM method and the optimistic case. Next, the LEAM model was tested for the "Mine and NOMBO" case considered in Kim's thesis. In this scenario, the P_{st} is lowered since detected NOMBOs must be avoided, increasing the sub's path length. Table 12 and Figure 34 display the results of this scenario. As expected, the LEAM model performed better than the MOAM model but below the optimistic (analytical upper bound) case.

Pd		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
LEAM Model Mine & NOMBOs	+.95 CI	0.197	0.214	0.250	0.280	0.332	0.395	0.450	0.532	0.637	0.794	1.000
	-.95 CI	0.137	0.154	0.190	0.220	0.272	0.335	0.390	0.472	0.577	0.734	0.970
	Mean	0.167	0.184	0.220	0.250	0.302	0.365	0.420	0.502	0.607	0.764	1.000
	Std Error	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
	# of Runs	594	642	733	801	901	991	1042	1064	1020	771	170
SMT Analytical Mean		**0.165**	**0.180**	**0.198**	**0.221**	**0.248**	**0.284**	**0.331**	**0.398**	**0.498**	**0.664**	**1.000**
MOAM Analytical Mean		**0.165**	**0.190**	**0.219**	**0.255**	**0.299**	**0.354**	**0.424**	**0.513**	**0.630**	**0.785**	**1.000**
Analytical Upper Bound		**0.165**	**0.198**	**0.237**	**0.284**	**0.340**	**0.407**	**0.487**	**0.583**	**0.698**	**0.835**	**1.000**

Table 12. Comparison of LEAM Simulation and SMT, MOAM, and Upper Bound ("Mine & NOMBO" Case).

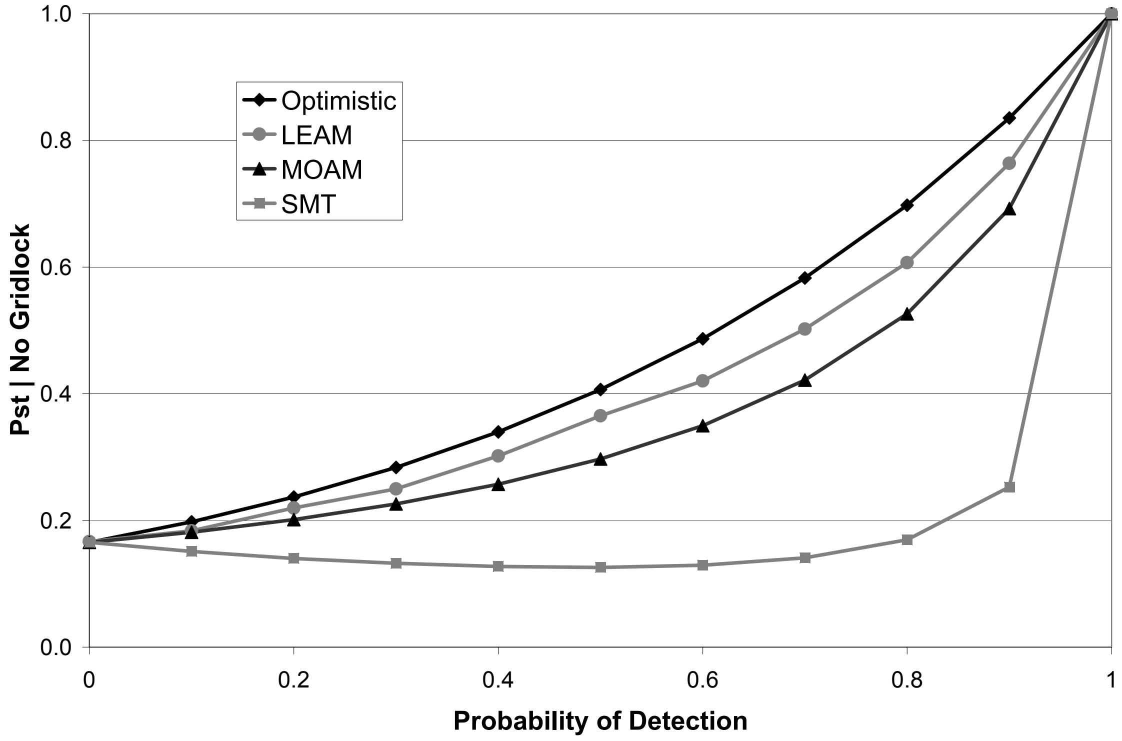


Figure 34. Comparison of Different Obstacle Avoidance Models ("Mine & NOMBO" Case).

3. New Scenarios

The only two scenarios considered so far have been somewhat unrealistic if considering a submarine transit of littoral chokepoints. Mines do not kill submarines at a range of 1000 yards as assumed in Table 13. More realistic lethal actuation ranges are 30 yards as discussed in Chapter IV. Instead of modeling the actuation range, further scenarios will use the minimum standoff distance developed in Chapter IV of 360 yards (assuming avoidance is based upon detection, not intelligence reports). In addition, the simulation reduced the minefield width to 20 nm (the approximate width of the Strait of

Hormuz). NOMBO densities are also much higher than the 0.3 per nm^2 as assumed in the previous scenario. Fortunately, most chokepoints and mined areas will only have a clutter category one as termed in Table 14. Still, up to 15 NOMBOs per nm^2 is considerable. The CNA study referred to in Chapter I also had a very low NOMBO rate. However, this was a function of the deep-water scenario that CNA considered. For shallow water chokepoint transits, a model must consider higher NOMBO densities. One recent analysis of a minefield used eight NOMBOs per nm^2.[21]

	Mine Only Case		Mine & NOMBO Case	
Parameters	Analytical Model	Simkit Simulation	Analytical Model	Simkit Simulation
mine density (mines/nm^2)	0.3	0.3	0.3	0.3
nombo density (nombos/nm^2)	0	0	0.3	0.3
w (twice actuation range) (nm)	1	1	1	1
L (nm)	6	6	6	6
Minefield Width (nm)	∞	100	∞	100

Table 13. Parameters for Previous Scenarios.

Mine densities are typically in the range of 1-5 mines per nm^2. Mines also come in different types, each with their own associated lethal range. A submarine transiting a mixed minefield would assume different lethal ranges for moored mines and bottom mines. The submarine would have to treat all detected bottom mines (or NOMBOs) as the highest threat bottom mine. The model assumes that the submarine is unable to distinguish between mines with a 2000-lb warhead or a 500-lb warhead. A 2000-lb warhead mine would be a typical value for worst-case bottom mine. Likewise, a typical value for worst-case moored mine would be a 1000-lb warhead. The SMT and MOAM models assumed that w was equal to twice the actuation range and that the submarine's sensor had a detection range equal to w. This model sets detection ranges of 1500 yards for moored mines and 750 yards for bottom mines. Table 15 summarizes the new scenario.

[21] Pollitt, George, Ian Craig, Joe Gezelter, Lance Hereford, "COMID 2002 Warfighter Payoff Analysis: Fusion Applications for Transit Deep Water SLOC," 29 August 2002.

Clutter Category	Nombos nm^2
1	<15
2	15-40
3	>40

Table 14. NOMBO Clutter Categories

Parameters	Previous Scenarios	New Scenarios
Moored mine density (mines/nm^2)	0.3	0-5
Bottom mine density (mines/nm^2)	N/A	0-5
NOMBO density (nombos/nm^2)	0 - 1	0 - 15
Moored mine standoff range (yds)	N/A	160
Bottom mine standoff range (yds)	N/A	160
w (twice actuation range) (yds)	1000	N/A
Sub's detection range of bottom mines (yds)	1000	750
Sub's detection range of moored mines (yds)	1000	1000
L (nm)	6	6
Minefield Width (nm)	100	20

Table 15. Comparing New Parameters with Previous Scenarios.

4. Exploration of LEAM Model

As expected, the LEAM model results had a P_{st} between the optimistic model and the MOAM model. Now that the previous section developed more realistic numbers to model, which scenarios should the LEAM simulation explore? The LEAM model should result in a P_{st} between the optimistic model and the MOAM model. Therefore, the simulation ran scenarios where this difference was significant.

Pd		0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
LEAM Model 5 Mines/nm^2 15 NOMBOs/nm^2	+.95 CI	0.038	0.035	0.037	0.091	0.104	0.110	0.128	0.213	0.356	0.542	1.000
	-.95 CI	-0.022	-0.025	-0.023	0.031	0.044	0.050	0.068	0.153	0.296	0.482	1.000
	Mean	0.008	0.005	0.007	0.061	0.074	0.080	0.098	0.183	0.326	0.512	1.000
	Std Error	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
	# of Runs	308	202	269	246	296	314	378	412	456	512	182
SMT Analytical Mean		**0.008**	**0.002**	**0.001**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**1.000**
MOAM Analytical Mean		**0.008**	**0.011**	**0.014**	**0.019**	**0.026**	**0.035**	**0.047**	**0.065**	**0.097**	**0.174**	**1.000**
Analytical Upper Bound		**0.008**	**0.013**	**0.021**	**0.035**	**0.056**	**0.091**	**0.147**	**0.237**	**0.383**	**0.619**	**1.000**

Table 16. Comparison of LEAM Simulation and SMT, MOAM, and Upper Bound ("Higher Densities, Smaller Actuation Range" Case).

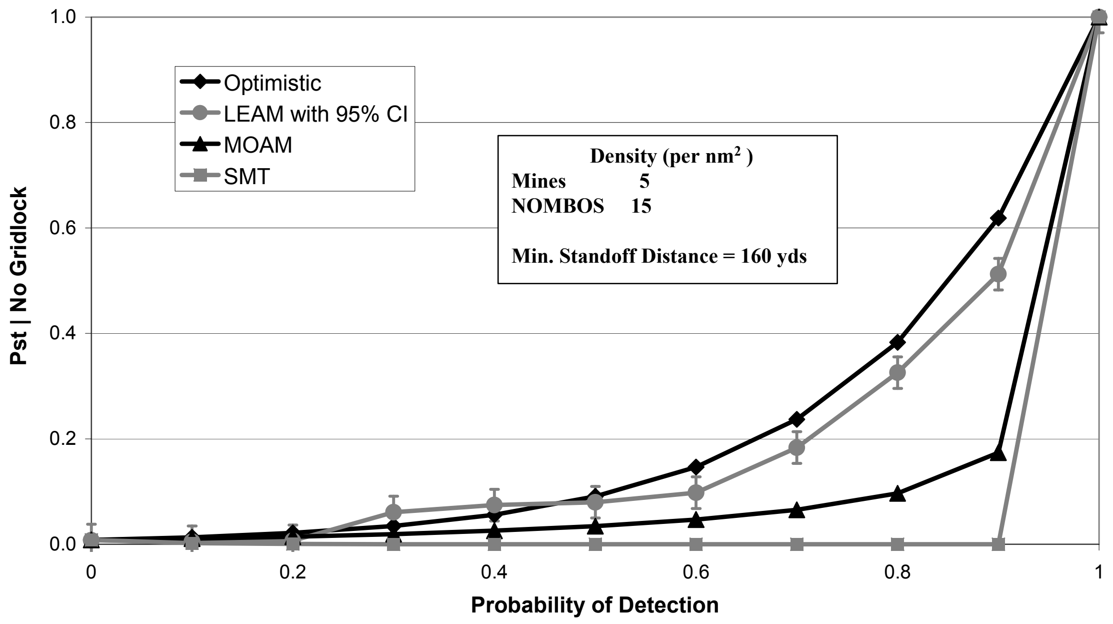


Figure 35. Comparison of Different Obstacle Avoidance Models (“Higher Densities, Smaller Actuation Range” Case).

As can be seen from Table 16 and Figure 35, the P_{st} for the LEAM model improves upon the previous models and comes close to the analytical upper bound. There is little margin left for future obstacle avoidance algorithms to improve upon. The next chapter uses the LEAM model to examine the problem of gridlock.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. GRIDLOCK

Avoiding mined areas will ensure the safety of submarines from mines, but it also will ensure that submarines are excluded from the theater of action…and allow the submarine force to become irrelevant.

"Mine Warfare and Submarines," Jim Crimmins, Proceedings

A. PROBLEM DEFINITION

If operators cannot distinguish between NOMBOs and mines, then they must give both mine-like contacts (NOMBOs and mines) the same minimum standoff distance. At some density level for mine-like contacts, a submarine's limited maneuverability will prevent it from penetrating a minefield. In Figure 36 for instance, not all of the detections need to be actual mines. Three factors increase the probability of gridlock. As mentioned earlier in Chapter I, as minimum standoff distance increases, the probability of gridlock increases because it becomes harder to find wide enough spaces to pass between mine-like contacts. Also, a narrower mined channel will increase the probability of gridlock because the narrower the channel, the less chances there are to find a clear path. Finally, as mine-like contact density increases, the probability of gridlock increases because if there are more mine-like contacts, on average, there will be less clear paths to find. In a real world problem, geography determines the chokepoint size; therefore, the simulation will keep this value fixed. The simulation also fixes the two values of minimum standoff distance (calculated in Chapter IV) since they based on factors known before entrance into a minefield. Therefore, the following example holds those two factors constant and examines the probability of gridlock as a function of mine-like contact (NOMBOs and mines) density.

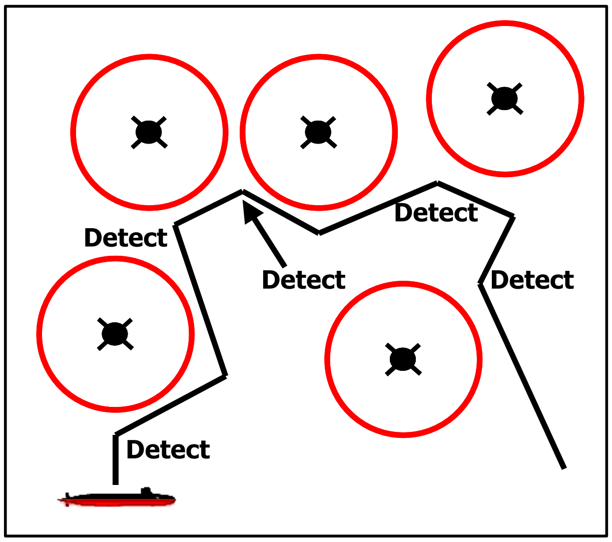


Figure 36. Submarine Experiencing Gridlock.

B. MODEL DESCRIPTION

The simulation in Chapter V focused on two possibilities for a minefield transit: survival or mine actuation. With a finite width minefield, the third possibility is gridlock. This chapter explores the same Lateral Excursion Avoidance Method (LEAM) model used in the previous simulation. The simulation in this chapter simply changes the parameters of interest and includes the possibility of gridlock. The common random number streams, Poisson generated minefield, and other aspects remain the same. Starting a submarine at the far southwest end of a minefield (assuming a northern destination), the LEAM model had a default avoidance direction of east. If the submarine's avoidance maneuver would place it past the eastern edge of the minefield, the simulation run would stop and report this as gridlock.

This analysis focuses on determining how object density physically changes the likelihood that clear paths exist through the minefield that could be found by the LEAM method. Accordingly, the simulation modeled the submarine with a perfect sensor (P_d = 1.0). Even with a perfect sensor, there are cases where the submarine cannot find any paths to get across the minefield. Using a perfect sensor ensured each replication determined if the minefield had a path or reached gridlock. The baseline scenario for gridlock simulation runs had a narrow mine line five nm wide and 1000 yards long with varying object density. With the spatial Poisson process assumption, adding densities of mines and detected NOMBOs (for P_d = 1, all NOMBOs are detected) provides an overall density of mine-like contacts that the submarine must avoid. The minimum standoff distance was set at 160 or 450 yards (based on Chapter IV).

C. ANALYSIS

The simulation increased mine-like contact density in increments of five per nm^2. For a minimum standoff distance of 160 yards, the first case of gridlock occurred at 55 mine-like contacts per nm^2, as shown in Table 17 and Figure 37. Mine density was again increased in increments of five per nm^2 until the probability of gridlock approached 1.0. Likewise, Table 18 and Figure 37 show that a minimum standoff distance of 450 yards first experienced gridlock at 11 mine-like contacts per nm^2. A similar analysis could be done for any chokepoint scenario of interest.

D. RESULTS

Object Density (per nm^2)	50	55	60	65	70	75	80	85	90	95	100	105	110	115	120	125	130	135	140	145	150
Prob Gridlock for 160 yard Min Standoff	0.00	0.03	0.03	0.11	0.21	0.23	0.28	0.39	0.50	0.59	0.65	0.83	0.87	0.88	0.88	0.90	0.92	0.94	0.97	0.97	1.00
Standard Error	0.01	0.03	0.03	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.03	0.02	0.03	0.03	0.03	0.01
# of Runs	100	100	150	250	250	250	250	250	250	250	250	250	250	250	250	150	100	100	100	100	100

Table 17. Gridlock Densities for 160-Yard Minimum Standoff Distance.

Object Density (per nm^2)	10	11	12	13	14	15	16	17	18	20	22	25	30	35	40	45	50	55	60	65
Prob Gridlock for 450 yard Min Standoff	0.00	0.03	0.03	0.04	0.10	0.14	0.16	0.20	0.28	0.42	0.56	0.67	0.73	0.91	0.96	0.96	0.98	0.99	0.99	1.00
Standard Error	0.01	0.02	0.02	0.02	0.03	0.03	0.04	0.04	0.04	0.05	0.05	0.05	0.04	0.03	0.02	0.02	0.02	0.01	0.01	0.01
# of Runs	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100

Table 18. Gridlock Densities for 450-Yard Minimum Standoff Distance.

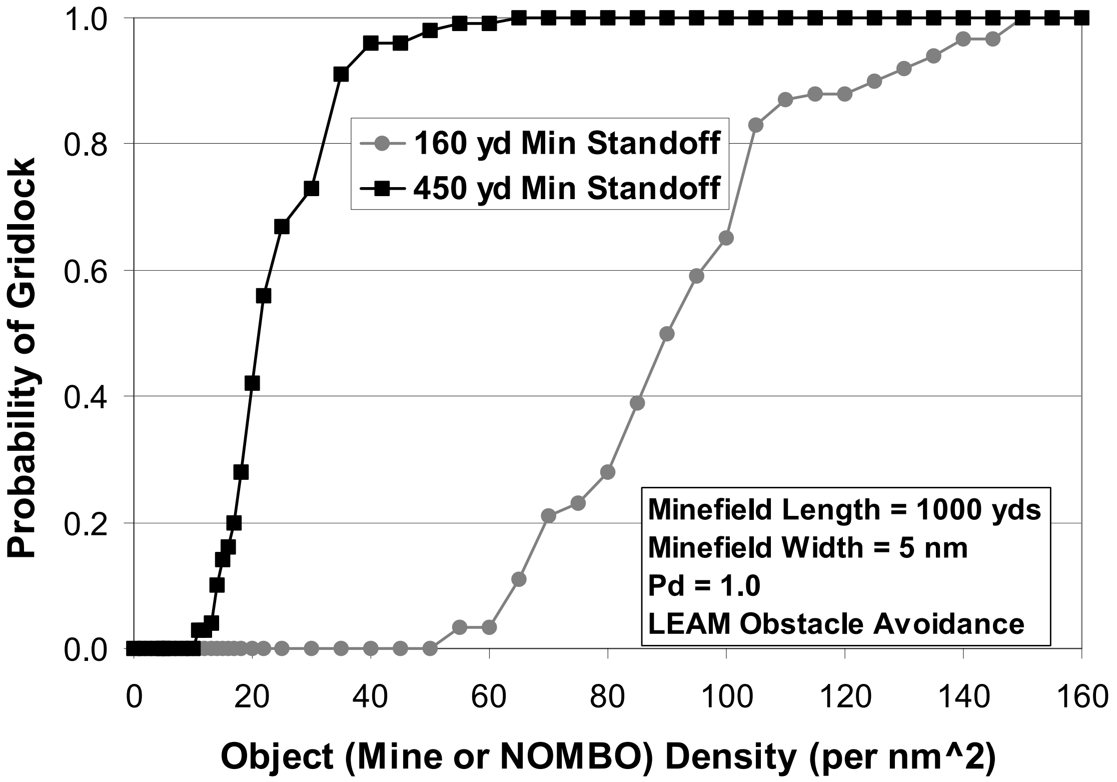


Figure 37. Gridlock Curves for Two Different Minimum Standoff Distances.

Not surprisingly, when a submarine increases its standoff distance, fewer mines are needed to cause gridlock. The military significance of this is noteworthy when considering mine reconnaissance. If another platform, such as an Unmanned Underwater Vehicle (UUV), surveyed a five nm wide chokepoint, the submarine could use this information when planning its route. However, the submarine would have to give a 450 yard (under the assumptions of Chapter IV) standoff distance to all of the mines reported by the UUV due to added mine position uncertainty and navigation error. With a 450-yard standoff distance, the five nm chokepoint with a mine density of roughly 21 mine-like contacts would have a 50% probability of gridlock. If the submarine were using its onboard sonar, thereby using a 160-yard standoff distance (from Chapter IV), a density of nearly 90 mine-like contacts would be required to give a 50% probability of gridlock. Therefore, under category II NOMBO densities (Table 14) of 15 to 40 NOMBOs per nm^2, the submarine could not transit the given chokepoint without having to rely on its own sensor (thereby reducing minimum standoff distance).

Notice that as standoff distance increases (from 160 to 450 yards), the curves in Figure 37 are approach a step-function. The curve for the 450-yard standoff distance is much steeper since one additional mine-like contact with a 900-yard diameter has a better chance of blocking the path than it would at the smaller standoff distance. There exists a parallel between the concept of gridlock and Percolation theory. Percolation theory concerns fluid flow in random media. Above a certain threshold, probability$_{\text{critical}}$ (p_c), the area contains a cluster that spans the entire space. According to Rinaldo B. Schinazi, below this threshold, "the origin is in a finite cluster with probability one."[22] As the space grows larger, the curve gets sharper until at infinity, the curve is a step function. Tim Frey and Ethan Decker state that this point is the "geometric analog of a phase transition."[23] The closer one gets to the phase transition, the more simulations runs are required.[24] Results from Percolation theory may provide insight to exploring the gridlock problem, and that is left for future work.

[22] Schinazi, Rinaldo B. Classical and Spatial Stochastic Processes. (Boston: Birkhauser, 1999): 113.

[23] Frey, Tim and Ethan Decker. "Percolation Theory." Ecological Complexity Seminar: Fall 1996, University of New Mexico, <http://sevilleta.unm.edu/~ehdecker/complexity/96fall/percol.html> [15 September 2003].

[24] "An Introduction to Percolation and Many-body Physics by Computer." http://fafnir.Phyast.pitt.edu/myjava/perc/pretest.html. [17 September 2003].

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND FUTURE WORK

The vulnerability of all classes of Navy ships to mine warfare is a neglected area of naval force planning.

2001 Naval Studies Board, Mine Warfare Assessment

A. CONCLUSIONS

Having a submarine penetrate a minefield is a very risky concept of operations. This thesis builds a foundation for future Tactical Decision Aids (TDA's).

The future use of battlespace preparation will require a model to maximize the probability of safe transit, such as the network model developed in this thesis. The safest route will not necessarily be the route with the lowest NOMBO density. Sometimes, the risk in the increased path length will outweigh the benefit of transiting the lowest NOMBO route. Although the particular parameters of the example (such as one nm^2 survey data) may change, future surveys could implement the network model.

The "distance into minefield" table serves as a baseline decision aid for choosing a distance to back up when encountering a mine. Given the assumptions and inputs, there is a 95% probability that the table's recommended distance will be enough for the submarine to back up out of the minefield. The table format allows the submarine commander a wide range of decision choices based on his trust of the inputs (sweep width and minefield density).

The calculations of minimum standoff distance highlight an important difference concerning mine avoidance. When detecting with an onboard sensor, the submarine's minimum standoff distance is much shorter than when avoiding a previously reported mine position. The submarine's navigation error that becomes a factor in the second case accounts for much of this difference.

Previous research had shown that as the level of detail in obstacle avoidance algorithms increased (SMT to MOAM models), the probability of safe transit increased. Likewise, the developed Lateral Excursion Avoidance Method (LEAM) shows higher probability of safe transits than either the SMT or MOAM model. The LEAM model comes close to the optimistic analytical model (the upper bound). In particular, the

LEAM model performed well in a scenario with realistic parameters for a littoral chokepoint transit.

Finally, even with a perfect sensor, a submarine can have its path blocked off and experience gridlock. An increase in minefield density or minimum standoff distance will increase the probability of gridlock. A decrease in minefield width will also increase the probability of gridlock. Further analysis could look more closely at the interaction affects of these three factors.

B. RECOMMENDED FOLLOW ON RESEARCH

1. Unmanned Undersea Vehicle

As mentioned in Chapter V, the use of an Unmanned Underwater Vehicle (UUV) to pre-survey the submarine route is a useful tool. This will certainly be the method of choice for minefield penetration. However, as mentioned the predictive path planning that the UUV allows can still revert to the reactive path planning necessary with an onboard sensor when previously undetected contacts appear. Another simulation could include the simulation from Chapter V as the reactive component of a submarine attempting to follow a path from a UUV route.

2. 3-D Environment

This project largely ignored the issue of depth. The mine threat was chosen based upon a shallow water (less than 300 feet) transit. The simulation's detection probabilities were based upon poor acoustic conditions expected in shallow water. However, choosing the proper search depth is also a critical role for a submarine penetrating a minefield. Further research could expand the simulation from Chapter V into a three dimensional environment.

Robotics research has developed the area of reactive planning and controlling autonomous vehicles. More can be done applying this research to the specific modeling of a submarine or UAV in a three dimensional minefield.

3. Concurrently Avoiding ASW Contacts

Minelayers sometimes lay mines for the express purpose of tying up an enemy's resources or redirecting them to an area more favorable to other assets. In World War II, the minefield across the English channel was designed to drive German submarines to the surface where they could be pursued by British patrol boats. Similar tactics today could pre-position ASW assets along a minefield's borders. As written in the second edition of Naval Operations Analysis, "It is also quite likely that the very sea lanes which offer the best opportunity for mine warfare are the very waters where the enemy concentrates his ASW forces."[25] The complex task of covertly penetrating such a minefield while simultaneously minimizing the risk of detection by the enemy ASW contacts would prove challenging. This potential threat certainly warrants more simulation and analysis.

4. Graphical User Interface and Clustering

The simulation from Chapter V provides helpful background research, but the submarine force still needs a TDA to keep better track of mine locations, NOMBO locations, cluster multiple contacts, suggest tracks, and provide other useful information. An eventual TDA could tie in multiple sources of information, use predictive path planning to determine the route based upon a UUV search, and reactive path planning to provide maneuvering options to the Officer of the Deck upon detecting a mine.

5. Multiple Injuries Leading to a Kill

In Chapter V, the simulation only considered the submarine killed if it exceeded a shock factor threshold. In the simulation, a submarine with a shock factor threshold of 0.5 for a kill could "safely" pass through the minefield and actuate 10 mines in a row receiving a shock factor of 0.49 each time. Assumptions that are more realistic would include the possibility of the submarine accumulating damage that could result in a kill. Future research should study the results of multiple injuries. Simulations could later include any physics based modeling or live testing results for further shock factor effects.

6. Exploring Battlespace Preparation

Any battlespace preparation plans must consider the tradeoffs between short paths and paths with less objects. Future work could explore the benefits of different survey block sizes when modeling a chokepoint. More closely integrating the Excel model from Chapter II and the simulation from Chapter V is another possibility for future work.

[25] Naval Operations Analysis. (2rd Ed. Annapolis: Naval Institute Press, 1977): 252.

7. Percolation Theory

Future research could benefit from investigating gridlock under the guise of Percolation theory. Rinaldo Schinazi's Classical and Spatial Stochastic Processes or texts on Statistical Physics are good starting points. In particular, Ronald Meester and Rahul Roy's Continuum Percolation gives a detailed presentation in solving Percolation in continuous problems (most other results are for a lattice model of the problem). This book defines a Poisson Boolean model $(\mathrm{X}, \rho, \lambda)$ with an "underlying Poisson point process X of density λ and radius random variable ρ." This translates well to the described gridlock problem when ρ is fixed.[26]

[26] Meester, Ronald and Rahul Roy. Continuum Percolation. (Cambridge University Press, Cambridge. 1996): 40.

APPENDIX A. DEFINITION OF TERMS

Advance: Distance gained in the direction of the original course until the ship steadies on its final course. It is measured from the point at which the rudder is put over.

BQS-14: High frequency sonar used by submarines for mine detection

CAC: Computer Aided Classification

CAD: Computer Aided Detection

CPA: Closest Point of Approach

Gridlock: At some density level for NOMBOs and mines, a submarine's limited maneuverability will prevent it from penetrating a minefield

Lateral Excursion Avoidance Method (LEAM): Obstacle avoidance algorithm implemented in Chapter V. Implemented in Simkit.

Mine Countermeasures Technical Advisory Group (MCMTAG): Formed by the director of the Submarine Warfare Division (N77) in May 2001 to look at MCM issues

Mine Detection and Avoidance (MDA): Concept of operations where a vessel transits across a known minefield. The vessel intends to successfully detect all contacts in its path and maneuver to avoid them so that it transits penetrates through the minefield safely.

Minimum Standoff Distance: The minimum distance in which to avoid actuating a mine of a given warhead size.

Minefield Object Avoidance Maneuver (MOAM): Second of two obstacle avoidance models implemented in Kim's thesis. Builds upon SMT model. Not implemented in Simkit.

NOn-mine Mine-like Bottom Object (NOMBO): Any object classified by the sonar operator as a mine that in reality is just some form of bottom clutter (old oil barrel, rock cropping, etc…). The key is that a NOMBO is indistinguishable from an actual mine and the penetrating ship must avoid the contact just as it would an actual mine since there is no way to classify between them.

Probability of Detection (P_d): The probability that a given sensor will detect a given object type.

Probability of Safe Transit (P_{st}): The probability that a boat will transit a given minefield without exceeding some damage threshold.

Set and Drift: Vectored effects of the ocean's current where set is the direction of the current and drift is the speed of the current.

Shock Factor (SF): A model of calculating the amount of shock felt by a vessel based on warhead weight, slant range, and depression angle (between vessel and mine). Vessels are often modeled as having a threshold value of SF that if exceeded will result in a kill.

Simkit: Package for developing discrete even models using Java 2. For more information, see http://diana.gl.nps.navy.mil/Simkit/

Simple Initial Threat (SIT): Probability that the first transit of a minefield results in a kill.

Simple Minefield Transit (SMT): First of two obstacle avoidance algorithms implemented in Kim's thesis. Implemented in Simkit as a baseline comparison with previous simulation and analytical model.

Tactical Decision Aid (TDA): A reference from as simple as a table to as complex as a computer program that provides a decision maker with information helpful in processing and deciding upon tactical decisions.

Transfer: The distance gained at right angles to the direction of the original course until the ship steadies on its final course.

UEP: Underwater Electric Potential: type of mine actuation mechanism

UUV: Unmanned Underwater Vehicle: autonomous vehicle that will pre-survey the minefield and relay that information to the submarine

APPENDIX B. SPATIAL POISSON PROCESS

This thesis frequently relied on the assumption that mines were distributed according to a spatial Poisson process. This appendix describes the spatial Poisson process for completeness. A brief digression into the general Poisson process will assist in the description. Poisson processes are frequently used in models and simulation since they are "highly amenable to analysis, and many results are known for them."[27] A Poisson process describes the number of events that occur in time t, N(t), when the rate is λ.

$$P\left(N(t)=n\right)=\frac{e^{-\lambda t}\left(\lambda t\right)^{n}}{n!} \qquad n = 0, 1, 2, \ldots \qquad \text{for } \lambda>0,\ t>0$$

Frequently, λ is given as a rate per unit time. For example, most queuing problems assume that the customer arrival process is Poisson. In that case, λ may refer to customers arriving at a certain rate per hour. Ross introduces an important theorem for the Poisson process:

> Given that N(t) = n, the n arrival times $S_1, \ldots, S_n$ have the same distribution as the order statistics corresponding to n independent random variables uniformly distributed on the interval (0,t).[28]

For example, assume customers are arriving at the rate of five per hour. Given that four customers are observed to arrive in one hour, the four arrival times between (0,1) are indistinguishable from four numbers drawn independently from a uniform distribution. The key to using this property of a Poisson process is that one first determines the Poisson number of arrivals, n. This theorem can be extended when considering events that occur in two-dimensions.

Taylor and Karlin define a spatial Poisson process for points N(A) in multi-dimensional region A if it meets the following conditions:

> (i) The numbers of points in nonoverlapping regions are independent random variables.

[27] Taylor, Howard M. and Samuel Karlin. An Introduction to Stochastic Modeling. 3rd Ed. San Diego: Academic Press, 1998: 313.

[28] Ross, Sheldon M., Introduction to Probability Models, 7th Ed., Academic Press, 2000: 272.

(ii) For any region A of finite volume, N(A) is Poisson distributed with mean $\lambda|A|$, where |A| is the volume of A.

Taylor and Karlin go on to define λ as a measure of intensity of the distribution that is independent of the size or shape. Some descriptions of the spatial Poisson process also refer to this λ as the density. For this thesis, λ was set as the mine density. Then, the Poisson distribution was used to generate the number of mines. Any mines generated were then distributed in x and y from independent uniform distributions. For example, let:

λ=0.3 mines/nm^2

Minefield width = 100 nm

Minefield length = 6 nm

Therefore, minefield area = 600 nm^2 and the mean number of mines would then be 600*0.3 = 180 mines. The Poisson distribution would be used to generate a random variate n, that represents the number of mines in a given replication. Then, those mines would be independently distributed at random (x,y) positions in the 100 nm by 6 nm rectangle, where each x is uniformly distributed between 0 and 100 and each y is uniformly and independently distributed between 0 and 16.

APPENDIX C: GAMS BATTLESPACE PREPARATION

```
$TITLE Battlespace Preparation for 5x5 area with 52 nodes & 115 arcs
$INLINECOM { }
 OPTIONS
   SOLPRINT =     OFF,
   DECIMALS =       1,
   LIMCOL   =   99999,
   LIMROW   =   99999,
   RESLIM   =      60, {max seconds}
   ITERLIM  =99999999, {max iterations}
   LP       =      XA,
   MIP      =      XA
 ;
 SETS
    n        "node"
 /
    n00*n51
 /
    arc(n,n)  "arc"
 /
n00.n01
n00.n02
n00.n03
n00.n04
n00.n05
n01.n06
n01.n10
n02.n06
n02.n07
n02.n11
n03.n07
n03.n08
n03.n12
n04.n08
n04.n09
n04.n13
n05.n09
n05.n14
n06.n10
n06.n11
n07.n11
n07.n12
n08.n12
n08.n13
```

n09.n13
n09.n14
n10.n15
n10.n19
n11.n15
n11.n16
n11.n20
n12.n16
n12.n17
n12.n21
n13.n17
n13.n18
n13.n22
n14.n18
n14.n23
n15.n19
n15.n20
n16.n20
n16.n21
n17.n21
n17.n22
n18.n22
n18.n23
n19.n24
n19.n28
n20.n24
n20.n25
n20.n29
n21.n25
n21.n26
n21.n30
n22.n26
n22.n27
n22.n31
n23.n27
n23.n32
n24.n28
n24.n29
n25.n29
n25.n30
n26.n30
n26.n31
n27.n31
n27.n32
n28.n33
n28.n37

```
n29.n33
n29.n34
n29.n38
n30.n34
n30.n35
n30.n39
n31.n35
n31.n36
n31.n40
n32.n36
n32.n41
n33.n37
n33.n38
n34.n38
n34.n39
n35.n39
n35.n40
n36.n40
n36.n41
n37.n42
n37.n46
n38.n42
n38.n43
n38.n47
n39.n43
n39.n44
n39.n48
n40.n44
n40.n45
n40.n49
n41.n45
n41.n50
n42.n46
n42.n47
n43.n47
n43.n48
n44.n48
n44.n49
n45.n49
n45.n50
n46.n51
n47.n51
n48.n51
n49.n51
n50.n51
 / ;
 alias(n,i,j) ;
 TABLE arcdata(i,j,*)
              cost
n00.n01       1.000
n00.n02       1.000
n00.n03       1.000
n00.n04       1.000
n00.n05       1.000
n01.n06       0.486
n01.n10       0.687
n02.n06       0.413
n02.n07       0.413
n02.n11       0.584
n03.n07       0.434
n03.n08       0.434
n03.n12       0.614
n04.n08       0.602
n04.n09       0.602
n04.n13       0.851
n05.n09       0.543
n05.n14       0.768
n06.n10       0.486
n06.n11       0.413
n07.n11       0.413
n07.n12       0.434
n08.n12       0.434
n08.n13       0.602
n09.n13       0.602
n09.n14       0.543
n10.n15       0.387
n10.n19       0.547
n11.n15       0.675
n11.n16       0.675
n11.n20       0.954
n12.n16       0.623
n12.n17       0.623
n12.n21       0.882
n13.n17       0.559
n13.n18       0.559
n13.n22       0.791
n14.n18       0.665
n14.n23       0.941
n15.n19       0.387
n15.n20       0.675
n16.n20       0.675
n16.n21       0.623
```

n17.n21	0.623
n17.n22	0.559
n18.n22	0.559
n18.n23	0.665
n19.n24	0.410
n19.n28	0.579
n20.n24	0.573
n20.n25	0.573
n20.n29	0.811
n21.n25	0.417
n21.n26	0.417
n21.n30	0.590
n22.n26	0.358
n22.n27	0.358
n22.n31	0.506
n23.n27	0.707
n23.n32	0.999
n24.n28	0.410
n24.n29	0.573
n25.n29	0.573
n25.n30	0.417
n26.n30	0.417
n26.n31	0.358
n27.n31	0.358
n27.n32	0.707
n28.n33	0.661
n28.n37	0.935
n29.n33	0.656
n29.n34	0.656
n29.n38	0.928
n30.n34	0.619
n30.n35	0.619
n30.n39	0.876
n31.n35	0.430
n31.n36	0.430
n31.n40	0.608
n32.n36	0.662
n32.n41	0.936
n33.n37	0.661
n33.n38	0.656
n34.n38	0.656
n34.n39	0.619
n35.n39	0.619
n35.n40	0.430
n36.n40	0.430
n36.n41	0.662

```
n37.n42     0.425
n37.n46     0.601
n38.n42     0.615
n38.n43     0.615
n38.n47     0.870
n39.n43     0.439
n39.n44     0.439
n39.n48     0.620
n40.n44     0.665
n40.n45     0.665
n40.n49     0.941
n41.n45     0.510
n41.n50     0.721
n42.n46     0.425
n42.n47     0.615
n43.n47     0.615
n43.n48     0.439
n44.n48     0.439
n44.n49     0.665
n45.n49     0.665
n45.n50     0.510
n46.n51     1.000
n47.n51     1.000
n48.n51     1.000
n49.n51     1.000
n50.n51     1.000
 ;
 VARIABLE
   TOTCOST          objective function value
 ;
 POSITIVE VARIABLES
   ASSIGN(i,j)      select arc as shortest s- path
 ;
 PARAMETERS
    LOGRISK(i,j)$arc(i,j) = ln(1-arcdata(i,j))
;
 EQUATIONS
   OBJECT
   NETFLOW(n)
 ;
 OBJECT..
   TOTCOST =e= SUM(arc(i,j),LOGRISK(i,j,'cost')*ASSIGN(i,j))
 ;
 NETFLOW(n)..
   SUM(arc(n,j),ASSIGN(n,j))   -   SUM(arc(i,n),ASSIGN(i,n))   =e=   -1   +
CARD(n)$(ORD(n)=1)
```

```
;
MODEL SHPATH
/
  OBJECT
  NETFLOW
/;
SOLVE SHPATH USING LP MINIMIZING TOTCOST ;
DISPLAY ASSIGN.l ;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. JAVA CODE FOR SIMKIT MODEL

A total of nine Java classes were written for Chapter V's simulation. First, instantiation of the *Mine*[29] class required a mover (holds position information and assigned a speed of zero), a sensor range, and a mine type. The M*ineType* class set constants for the types of mines considered. For programming purposes, NOMBOs were grouped together with mines, however all NOMBOs were given a sensor range of zero. The *PoissonMinefield* class took in the minefield's width and length, densities of all mine types, and actuation ranges for bottom and moored mines. Then, it calculated out the number of mines (from a Poisson distribution) based upon the assigned density and minefield area. *PoissonMinefield* then instantiated the resulting number of mines and placed them using the *RandomPointGenerator* class. The *ProbCookieCutterMediator* class was an extension of Simkit's cookie cutter mediator. The probability cookie cutter mediator took in a probability of detection. In Simkit, a referee knows the ground truth for all movers and sensors (in this case the submarine and the mines). The referee would inform the mediator when a mine had entered the range of the submarine's sensor. Then the mediator would draw a uniform random number and compare it to the probability of detection associated with that particular sensor. If the uniform draw was less than the probability of detection, then a "detection" event was immediately scheduled. The *SubmarineSMT* class and the *SubmarineLEAM* class contained the obstacle avoidance logic discussed in Chapter V. Finally, the main class, *SimulateSubMDA*, instantiates the Poisson minefield, the submarine (whichever obstacle avoidance was tested), and assigns the referee and mediators for the discrete event simulation. All classes used in Chapter V's simulation interact with the Simkit package for Discrete Event Simulation. Simkit is available for download[30] and has available documentation online.[31]

[29] Note: All class names in this appendix follow the Java formatting rules and are indicated in italics.

[30] http://diana.gl.nps.navy.mil/Simkit/simkit_1.2.8/install.htm

[31] http://diana.gl.nps.navy.mil/Simkit/doc/

A. MINE

```
/*
 * Mine.java
 * Created on August 18, 2003, 8:57 AM
 */

package submda;
import simkit.smdx.*;
import java.awt.geom.*;

public class Mine extends simkit.smdx.CookieCutterSensor {

   private MineType myType;
   protected boolean actuated;

   /** Creates a new instance of Mine */
   public Mine(Mover mover, double range, MineType type) {
      super(mover, range);
      myType = type;
   }

   public MineType getType() { return myType; }

   public void reset() {
      super.reset();
      actuated = false;
   }

   public String toString() {
      return myType + " " + getLocation();
   }

   public void doDetection(Moveable m) {
      waitDelay("Actuation", 0.0);
   }
   public void doActuation() {
     actuated = true;
     firePropertyChange("actuated", Boolean.FALSE, Boolean.TRUE);
   }

   public void doUndetection(Moveable m) {
   }

   public boolean isActuated() { return actuated; }

}
```

B. MINE TYPE

```
/*
 * MineType.java
 * Created on August 18, 2003, 8:55 AM
 */

package submda;

public class MineType {

   public static final MineType MOORED = new MineType("Moored");
   public static final MineType BOTTOM = new MineType("Bottom");
   public static final MineType NOMBO = new MineType("NOMBO");

   private String name;

   /** Creates a new instance of MineType */
   protected MineType(String name) {
      this.name = name;
   }

   public String toString() {
      return name;
   }
}
```

C. POISSON MINEFIELD

```
/*
 * PoissonMinefield.java
 * Created on August 17, 2003, 1:12 PM
 * Terry Nawara
 */

package submda;
import java.util.*;
import simkit.*;
import simkit.random.*;
import simkit.smdx.*;
import java.awt.geom.*;
import java.text.*;

public class PoissonMinefield extends SimEntityBase{

   // OUTLINE of code
   // Instance Variables (parameters and state variables)
   // Constructor
   // Instance Methods - Getters (for parameters and state variables)
   // Instance Methods - Setters (for parameters only)
   // Instance Methods - Other (reset, toString...)

   // Instance Variables
   private int counter = 0;    // used for statistics to get total # of mines actuated
   private DiscreteRandomVariate randMoor;
   private DiscreteRandomVariate randBott;
   private DiscreteRandomVariate randNombo;

   // Minefield:   Parameters (initialized in constructor)
   private double widthNm;
   private double lengthNm;
   private double [ ] corners;  // Corners used for call to RandomPointGenerator
   private RandomPointGenerator rpg;
   private double areaNm;
   private MineType MOORED = new MineType("Moored");
   private MineType BOTTOM = new MineType("Bottom");
   private MineType NOMBO = new MineType("NOMBO");

   // Mines:   Parameters (initialized in constructor)
   private double moorMineDensity;  // mines per square nm
   private double bottMineDensity;  // mines per square nm
   private double globalMoorMineActuationRangeYards;
   private double globalBottMineActuationRangeYards;
   private Mine [ ] moorMineList = new Mine[0];
   private Mine [ ] bottMineList = new Mine[0];

   // NOMBOs (NOn-mine Minelike Bottom Objects):  Parameters (initialized in
                   constructor)
   private double nomboDensity;    // NOMBOs per square nm
   private int totalNombos;
   private Mine [ ] nomboList = new Mine[0];

   // Constructor
   /** Creates a new instance of Minefield */
   public PoissonMinefield(double width, double length, double moorDensity, double
             bottDensity, double nomboDensity, double moorRange, double bottRange) {
     this.widthNm = width;
     this.lengthNm = length;
     this.moorMineDensity = moorDensity;
     this.bottMineDensity = bottDensity;
     this.nomboDensity = nomboDensity;
     this.globalMoorMineActuationRangeYards = moorRange;
     this.globalBottMineActuationRangeYards = bottRange;

     // Following sets up random point generator
     corners = new double[ ] {0.0, 0.0, (width*2000.0),length*2000.0};
     rpg = new RandomPointGenerator(corners);

     // Following defines area
     areaNm = widthNm*lengthNm;

     // Set Number of mine like contacts on given densities according to a Poisson Process
     if (moorMineDensity>0.0) {
         this.randMoor= (DiscreteRandomVariate)
         RandomVariateFactory.getInstance("submda.PoissonVariate", new Object[ ]
             {new Double(moorMineDensity*areaNm)} );
     }
     if (bottMineDensity>0.0) {
         this.randBott= (DiscreteRandomVariate)
         RandomVariateFactory.getInstance("submda.PoissonVariate", new Object[ ]
            {new Double(bottMineDensity*areaNm)},
         randMoor.getRandomNumber());
     }
     if (nomboDensity>0.0) {
         this.randNombo= (DiscreteRandomVariate)
         RandomVariateFactory.getInstance("submda.PoissonVariate", new Object[ ]
            {new Double(nomboDensity*areaNm)},
          randMoor.getRandomNumber());
     }
   }
```

```
}


 //Instance Methods - Getter Methods (for states and parameters)
public double getWidthNm() {
  return widthNm;
}
public double getLengthNm() {
  return lengthNm;
}
public double getAreaNm() {
  return areaNm;
}
public double getMoorMineDensity() {
  return moorMineDensity;
}
public double getBottMineDensity() {
  return bottMineDensity;
}
public int getTotalMoorMines() {
  return moorMineList.length;
}
public int getTotalBottMines() {
  return bottMineList.length;
}
public int getTotalMines() {
  return  (moorMineList.length + bottMineList.length);
}
public double getGlobalMoorMineActuationRangeYards() {
  return globalMoorMineActuationRangeYards;
}
public double getGlobalBottMineActuationRangeYards() {
  return globalBottMineActuationRangeYards;
}
public double getNomboDensity() {
  return nomboDensity;
}
public int getTotalNombos() {
  return nomboList.length;
}
public Point2D getBottMineLocation(int i) {
  return bottMineList[i].getLocation();
}
public Point2D getMoorMineLocation(int i) {
  return moorMineList[i].getLocation();
}
public Point2D getNomboLocation(int i) {
  return nomboList[i].getLocation();
}
```

```
public Mine[ ] getMoorMineList() {
  return moorMineList;
}
public Mine[ ] getBottMineList() {
  return bottMineList;
}
public Mine[ ] getNomboList() {
  return nomboList;
}
public Mine getMoorMine(int i) {
  return moorMineList[i];
}
public Mine getBottMine(int i) {
  return bottMineList[i];
}
public Mine getNombo(int i) {
  return nomboList[i];
}
public int getTotalNumberMoorMinesActuated() {
  counter = 0;
  for (int i=0; i< moorMineList.length; i++) {
    if(moorMineList[i].isActuated()) {
      counter ++;
    }
  }
  return counter;
}
public int getTotalNumberBottMinesActuated() {
  counter = 0;
  for (int i=0; i< bottMineList.length; i++) {
    if (bottMineList[i].isActuated()) {
      counter ++;
    }
  }
  return counter;
}

//Instance Methods - Setter Methods:  for parameters only (not for state variables)
// Note:  No setter method for length & width.  To do this must reinstate another
//  minefield.  length & width effect the area, the # of mines, the corners, etc...
public void setMoorMineDensity(double number) {
  moorMineDensity = number;
  randMoor= (DiscreteRandomVariate) RandomVariateFactory.getInstance("Poisson",
         new Object[ ] {new Double(moorMineDensity*areaNm)} );
}
```

```
public void setBottMineDensity(double number) {
  bottMineDensity = number;
  randBott= (DiscreteRandomVariate) RandomVariateFactory.getInstance("Poisson",
         new Object[ ] {new Double(bottMineDensity*areaNm)} );
}
public void setGlobalMoorMineActuationRangeYards(double number) {
  globalMoorMineActuationRangeYards = number;
}
public void setGlobalBottMineActuationRangeYards(double number) {
  globalBottMineActuationRangeYards = number;
}
public void setNomboDensity(double number) {
  nomboDensity = number;
  randNombo= (DiscreteRandomVariate) RandomVariateFactory.getInstance
         ("Poisson", new Object[ ] {new Double(nomboDensity*areaNm)} );
}

// Instance Methods - other than Getter, Setter, or "do"
// restore state variable to their initial values
// lay new minefield - don't reset random variables!
  public void reset() {
  super.reset();

  // If there are already moored mines, remove them from schedule
  if (moorMineList != null) {
    for (int i = 0; i < moorMineList.length; ++i) {
      Schedule.removeRerun(moorMineList[i]);
      Schedule.removeRerun(moorMineList[i].getMover());
    }
  }
  // Instantiate moored mines
  if (moorMineDensity>0.0) {
    moorMineList = new Mine[randMoor.generateInt()];
    for (int i=0; i< moorMineList.length; i++) {
      String moorMineName = "Moored Mine "+ (i+1);
      moorMineList[i] = new Mine(new UniformLinearMover(moorMineName,
         rpg.generatePoint(), 0.0), globalMoorMineActuationRangeYards, MOORED);
    }
  }
  // If there are already bottom mines, remove them from schedule
  if (bottMineList != null) {
    for (int i = 0; i < bottMineList.length; ++i) {
      Schedule.removeRerun(bottMineList[i]);
      Schedule.removeRerun(bottMineList[i].getMover());
    }
  }
```

```
    // Instantiate bottom mines
    if (bottMineDensity>0.0) {
      bottMineList = new Mine[randBott.generateInt()];
      for (int i=0; i< bottMineList.length; i++) {
        String bottMineName = "Bottom Mine "+ (i+1);
        bottMineList[i] = new Mine(new UniformLinearMover(bottMineName,
           rpg.generatePoint(), 0.0), globalBottMineActuationRangeYards, BOTTOM);
      }
    }
    // If there are already nombos, remove them from schedule
    if (nomboList != null) {
      for (int i = 0; i < nomboList.length; ++i) {
        Schedule.removeRerun(nomboList[i]);
        Schedule.removeRerun(nomboList[i].getMover());
      }
    }
    // Instantiate nombos
    if (nomboDensity>0.0) {
      nomboList = new Mine[randNombo.generateInt()];
      for (int i=0; i< nomboList.length; i++) {
        String nomboName = "Nombo "+ (i+1);
        nomboList[i] = new Mine(new UniformLinearMover(nomboName,
           rpg.generatePoint(), 0.0), 0.0, NOMBO);
      }
    }

  }

  public String paramString() {
    return "POISSON MINEFIELD"  + '\n'
    + "Measuring " + widthNm + " miles wide and " + lengthNm + " miles long.  Total
area = " + areaNm + " (nm^2)" + '\n' + '\n'
    + "Number of mine-like contacts            Density (per nm^2) " + '\n'
    + "     NOMBOs      " + nomboList.length +      "               " + nomboDensity + '\n' + '\n'
    + "     Moored Mines  " + moorMineList.length +  "            " + moorMineDensity + \n'
    + "     Bottom Mines  " + bottMineList.length +  "              " + bottMineDensity +'\n'
    + "     TOTAL Mines = " + (moorMineList.length+bottMineList.length) + '\n' + '\n'
    + "Global Mine Actuation Ranges (yards) " + '\n'
    + "     Moored Mines = " + globalMoorMineActuationRangeYards + '\n'
    + "     Bottom Mines = " + globalBottMineActuationRangeYards + '\n' + '\n'
    + "Moored Mine Locations: " + '\n' + getMineArrayAsString(moorMineList) + '\n'
    + "Bottom Mine Locations: " + '\n' + getMineArrayAsString(bottMineList) + '\n';
  }

  public static String getMineArrayAsString(Mine[ ] mine) {
    DecimalFormat form = new DecimalFormat("0.00");
    StringBuffer buf = new StringBuffer();
    for (int i = 0; i < mine.length; ++i) {
      Point2D loc = mine[i].getLocation();
      buf.append('[');
      buf.append(form.format(loc.getX()));
      buf.append(',');
      buf.append(form.format(loc.getY()));
      buf.append(']');
      buf.append(' ');
      if ( (i + 1) % 1 == 0) { buf.append('\n'); }
    }
    return buf.toString();
  }

}
```

D. RANDOM POINT GENERATOR

```
package submda;

import simkit.random.*;
import java.awt.geom.*;

// Used by PoissonMinefield to generate Point2D

public class RandomPointGenerator {

   private RandomVariate[ ] rv;

   public RandomPointGenerator(Point2D[ ] cornerPts) {
      this(new double[ ] {cornerPts[0].getX(), cornerPts[0].getY(),
         cornerPts[1].getX(), cornerPts[1].getY()});
   }

   public RandomPointGenerator(double[ ] corners) {
      if (corners.length != 4) {
         throw new IllegalArgumentException("Need 4 corners: " + corners.length);
      }
      RandomVariate[ ] rand = new RandomVariate[2];
      rand[0] = RandomVariateFactory.getInstance("Uniform",
         new Object[ ] { new Double(corners[0]), new Double(corners[2]) });
      rand[1] = RandomVariateFactory.getInstance("Uniform",
         new Object[ ] { new Double(corners[1]), new Double(corners[3]) },
         rand[0].getRandomNumber());
      setRandomVariate(rand);
   }

   public void setSeed(long seed) {
      rv[0].getRandomNumber().setSeed(seed);
      rv[1].setRandomNumber(rv[0].getRandomNumber());
   }

   /** Creates a new instance of RandomPointGenerator */
   public RandomPointGenerator(RandomVariate[ ] rv) {
      setRandomVariate(rv);
   }

   public void setRandomVariate(RandomVariate[ ] rv) {
      if (rv.length != 2) {
         throw new IllegalArgumentException("Need array of length 2: " + rv.length);
      }
      this.rv = (RandomVariate[ ]) rv.clone();
   }
   public RandomVariate[ ] getRandomVariate() {
        return (RandomVariate[ ]) rv.clone();
   }
   public Point2D.Double generatePoint() {
      return new Point2D.Double(rv[0].generate(), rv[1].generate());
   }
   public String toString() {
      return "RandomPointGenerator " + rv[0] + " - " + rv[1];
   }
    public static void main(String[ ] args) {
      double[ ] c = new double[ ] { 0, 0, 50, 250 };
      RandomPointGenerator rpg = new RandomPointGenerator(c);
      System.out.println(rpg);
      for (int i = 0; i < 5; ++i) {
         System.out.println(rpg.generatePoint());
      }

      Point2D[ ] pts = new Point2D[2];
      pts[0] = new Point2D.Double(0.0, 0.0);
      pts[1] = new Point2D.Double(30.0, 350.0);

      rpg = new RandomPointGenerator(pts);
      System.out.println(rpg);
      for (int i = 0; i < 5; ++i) {
         System.out.println(rpg.generatePoint());
      }

   }

}
```

E. PROBABILITY COOKIE CUTTER SENSOR

```
/*
 * ProbCookieCutterSensor.java
 * Created on August 19, 2003, 10:58 PM
 * Terry Nawara
 * version 1.2
 */

package submda;
import simkit.*;
import simkit.smdx.*;

/** A sensor whose probability of detection is a given constant.
 * After the target has entered its range, a random uniform is compared to the
 * prob of detection to determine whether the sensor detects the target.
  */
public class ProbCookieCutterSensor extends CookieCutterSensor implements Sensor  {

   // Instance Variables
   private double probDetect;

   // Constructor
   public ProbCookieCutterSensor(double range, Mover mover, double prob) {
      super(range, mover);
      setProbDetect(prob);
   }
   // Instance Method - Setter
   public void setProbDetect(double probability) {
      probDetect = probability;
   }
   // Instance Method - Getter
   public double getProbDetect() {
      return probDetect;
   }

   public void doUndetection(Moveable contact) {
      if (getContacts().contains(contact)) {
         super.doUndetection(contact);
      }
   }

   // Instance Method
   public String toString() {
       return "ProbCookieCutterSensor" + super.toString().substring(18);
   }
}
```

F. PROBABILITY COOOKIE CUTTER MEDIATOR

```
/*
 * ProbCookieCutterMediator.java
 * Created on August 19, 2003, 11:07 PM
 */

package submda;
import simkit.*;
import simkit.smdx.*;
import simkit.random.*;
import java.beans.*;
import java.util.*;

public class ProbCookieCutterMediator extends SimEntityBase implements
SensorTargetMediator{

   // Class Variables
   public static RandomNumber uniformDraw;

   // Instance Variables
   private WeakHashMap contacts;

   // Static Constructor
   static {
      uniformDraw = RandomNumberFactory.getInstance();
   }

   // Class Methods
   public static void setSeed(long seed) {
      uniformDraw.setSeed(seed);
   }

   // Constructor
   public ProbCookieCutterMediator() {
      contacts = new WeakHashMap();
   }

   // Instance Methods
   public void doEnterRange(Sensor sensor, Mover mover) {
      if (sensor instanceof ProbCookieCutterSensor) {
         ProbCookieCutterContact contact = (ProbCookieCutterContact)
            contacts.get(mover);
         if (contact == null) {
            contact = new ProbCookieCutterContact(mover);
            contacts.put(mover, contact);
         }
```

```
            if (uniformDraw.draw()<((ProbCookieCutterSensor)sensor).getProbDetect()) {
               sensor.waitDelay("Detection", 0.0, new Object[] { contact });
            }
         }
      }

      public void doExitRange(Sensor sensor, Mover mover) {
         if (sensor instanceof ProbCookieCutterSensor) {
            Object[] contact = new Object[] { contacts.get(mover) };
            sensor.waitDelay("Undetection", 0.0, contact);
         }
      }

      public void propertyChange(PropertyChangeEvent e)
      {
      }
   }
```

G. SUBMARINE SMT

```
package submda;

/*
 * SubmarineSMT.java
 * Created on June 9, 2003, 9:50 AM
 * Terry Nawara
 */

import java.util.*;
import simkit.*;
import simkit.random.*;
import simkit.smdx.*;
import java.awt.geom.*;
import java.text.DecimalFormat;

// Uses Simple Minefield Transit (SMT) for Obstacle Avoidance

public class SubmarineSMT extends SimEntityBase{

  // OUTLINE of code
  // Instance Variables (parameters and state variables)
  // Constructor
  // Instance Methods - Getters (for parameters and state variables)
  // Instance Methods - Setters (for parameters only)
  // Instance Methods - doEvents (doRun, doDetection...)
  // Instance Methods - Other (reset, toString...)

  // Instance Variables
  DecimalFormat twoPlaces = new DecimalFormat("0.00");

  // Local Input Variables for Submarine:   Parameters (initialized in constructor)
  private Point2D.Double subStart;
  private Point2D.Double subFinish;
  private double newStart;           // used to set reentry point
  private double subMaxSpeedKnots;
  private double subTransitSpeedKnots;
  private double meanTimeSubDetectContact;    // assuming exponetial; units of hours
          (constructor takes input in minutes and coverts units) // not used in this version
  private double subDetectMoorMineRangeYards;
  private double subDetectBottMineRangeYards;
  private double subDetectNomboRangeYards;    // (should be similar to range for bottom
          mines)
  private Sensor subSonarMoored;
  private Sensor subSonarBottom;
  private Sensor subSonarNombo;
private Mover subMover;
private PathMoverManager subMoverManager;

private double minefieldWidthYds;
private double minefieldLengthYds;

// Local Input Variables for Submarine:   State Variables (initialized in setter methods)
private int numberDetections = 0;   // used to count firePropertyChange upon detections
private double amountTraveled = minefieldLengthYds;  // used to count total travel
private boolean gridlock = false;  // only turns true when the submarine has tried to
        cross all minefield lanes and has found a mine each time

// Constructor
/** Creates a new instance of Sub */
public SubmarineSMT (Point2D.Double start, Point2D.Double finish, double
        maxSpeed, double transitSpeed, double time, double detMoorRange, double
        detBottRange, double detNomboRange, double width, double length, double
        prob) {
  this.subStart = start;
  this.subFinish = finish;
  this.subMaxSpeedKnots = maxSpeed;
  this.subTransitSpeedKnots = transitSpeed;
  this.meanTimeSubDetectContact = time/60.0;              // not used in this version!
  this.subDetectMoorMineRangeYards = detMoorRange;
  this.subDetectBottMineRangeYards = detBottRange;
  this.subDetectNomboRangeYards = detNomboRange;
  this.minefieldWidthYds = width*2000.0; // convert to yards
  this.minefieldLengthYds = length*2000.0; // convert to yards
  this.probDetect = prob;

  subMover = new UniformLinearMover("Sub", subStart,
        subMaxSpeedKnots*2000.0);    // mover is in units yds/hour not kts
  subMoverManager = new PathMoverManager(subMover, new ArrayList());
  subMoverManager.setStartOnReset(true);
  subMoverManager.addWayPoint(subFinish, subTransitSpeedKnots*2000.0);

    /*
   subSonarMoored = new ConstantRateSensor(subDetectMoorMineRangeYards,
        subMover, meanTimeSubDetectContact);
   subSonarBottom = new ConstantRateSensor(subDetectBottMineRangeYards,
        subMover, meanTimeSubDetectContact);
   subSonarNombo= new ConstantRateSensor(subDetectNomboRangeYards,
        subMover, meanTimeSubDetectContact);
    */

  subSonarMoored = new ProbCookieCutterSensor(subDetectMoorMineRangeYards,
          subMover, probDetect);
    subSonarBottom = new ProbCookieCutterSensor(subDetectBottMineRangeYards,
          subMover, probDetect);
    subSonarNombo= new ProbCookieCutterSensor(subDetectNomboRangeYards,
          subMover, probDetect);

    // Submarine listens to all sensors (moored, bottom, and nombo sonars)
    subSonarMoored.addSimEventListener(this);
    subSonarBottom.addSimEventListener(this);
    subSonarNombo.addSimEventListener(this);
  }

  // Instance Method - Getter Methods (for states and parameters)
  public Point2D.Double getSubStart() {
    return subStart;
  }
  public Point2D.Double getSubFinish() {
    return subFinish;
  }
  public double getSubMaxSpeedKnots() {
    return subMaxSpeedKnots;
  }
  public double getSubTransitSpeedKnots() {
    return subTransitSpeedKnots;
  }
  public double getMeanTimeSubDetectContact() {
    return meanTimeSubDetectContact;
  }
  public double getSubDetectMoorMineRangeYards() {
    return subDetectMoorMineRangeYards;
  }
  public double getSubDetectBottMineRangeYards() {
    return subDetectBottMineRangeYards;
  }
  public double getSubDetectNomboRangeYards() {
    return subDetectNomboRangeYards;
  }
  public Sensor getSubSonarBottom() {
    return subSonarBottom;
  }
  public Sensor getSubSonarMoored() {
    return subSonarMoored;
  }
  public Sensor getSubSonarNombo() {
    return subSonarNombo;
  }
```

```
public Mover getSubMover() {
  return subMover;
}
public PathMoverManager getSubMoverManager() {
  return subMoverManager;
}
public int getNumberDetections() {
  return numberDetections;
}
public double getAmountTraveled() {
  return amountTraveled;
}
public double getProbDetect() {
  return probDetect;
}
public boolean isGridlock() {
  return gridlock;
}

// Instance Method - Setter Methods:  for parameters only (not for state variables)
public void setSubStart(Point2D.Double point) {
  subStart = point;
}
public void setSubFinish(Point2D.Double point) {
  subFinish = point;
}
public void setSubMaxSpeedKnots(double speed) {
  subMaxSpeedKnots = speed;
}
public void  setSubTransitSpeedKnots(double speed) {
  subTransitSpeedKnots = speed;
}
public void setMeanTimeSubDetectContact(double time) {
  meanTimeSubDetectContact = time;
}
public void setSubDetectMoorMineRangeYards(double range) {
  subDetectMoorMineRangeYards = range;
}
public void setSubDetectBottMineRangeYards(double range) {
  subDetectBottMineRangeYards = range;
}
public void setSubDetectNomboRangeYards(double range) {
  subDetectNomboRangeYards = range;
}
public void setProbDetect(double prob) {
  ((ProbCookieCutterSensor)subSonarMoored).setProbDetect(prob);
  ((ProbCookieCutterSensor)subSonarBottom).setProbDetect(prob);
  ((ProbCookieCutterSensor)subSonarNombo).setProbDetect(prob);
}

// Instance Methods - doEvents (doRun, doDetection...)

public void doRun() {
}

public void doDetection(Moveable contact) {
  firePropertyChange("numberDetections", numberDetections, numberDetections++);
  amountTraveled = amountTraveled + 2.0*subMover.getLocation().getY();
  if(minefieldWidthYds-subMover.getLocation().getX() < 2000.0) {
    waitDelay("doGridlock",0.0);
  }
  else{
    subMoverManager.stop();
    subMoverManager.clearPath();
    // Goes back with a buffer zone in case some mines are right along the front
    subMoverManager.addWayPoint(new Point2D.Double(subMover.getLocation().
         getX(),  -2000.0), subTransitSpeedKnots*2000.0);
    newStart = subMover.getLocation().getX() + 2.0*subDetectMoorMineRangeYards;
         // hbar = w
    subMoverManager.addWayPoint(new Point2D.Double  (newStart, -2000.0),
         subTransitSpeedKnots*2000.0);
    subMoverManager.addWayPoint(new Point2D.Double  (newStart,
         minefieldLengthYds), subTransitSpeedKnots*2000.0);
    subMoverManager.start();
  }
}
public void doGridlock() {        // effectively stops simulation since no more events
         scheduled after sub returns to bottom of minefield
  gridlock = true;
  firePropertyChange("gridlock", Boolean.FALSE, Boolean.TRUE);
  subMoverManager.stop();
  subMoverManager.clearPath();
  subMoverManager.addWayPoint(new Point2D.Double(subMover.getLocation().
         getX(), -2000.0), subTransitSpeedKnots*2000.0);
  subMoverManager.start();
}

// Instance Methods - other than Getter, Setter, or "do"
// restore state variable to their initial values
 public void reset() {
  super.reset();
  subMoverManager.clearPath();
```

```
    subMoverManager.addWayPoint(subFinish, subTransitSpeedKnots*2000.0);
    numberDetections = 0;
    amountTraveled = minefieldLengthYds;
  }

  public String paramString() {
    return "SUBMARINE" + '\n'
    + "Sub's Start Position  = (" + twoPlaces.format(subStart.getX()) + ", " +
twoPlaces.format(subStart.getY()) + ") units of yards" +'\n'          // Try to show in units
of nm instead
    + "Sub's Finish Objective = (" + twoPlaces.format(subFinish.getX()) + ", " +
twoPlaces.format(subFinish.getY()) + ") units of yards" +'\n' + '\n'  // Try to show in units
of nm instead
    + "Sub's Max Speed (knots)     = " + subMaxSpeedKnots + '\n'
    + "Sub's Transit Speed (knots) = " + subTransitSpeedKnots + '\n'
    + "Sub's mean time to detect mine (minutes) = " + meanTimeSubDetectContact*60.0
+ '\n'
    + "Sub's probability of detection = " + probDetect + '\n'
    + "Sub's detection range (yards) for"  + '\n'
    + "     Moored Mines =  " + subDetectMoorMineRangeYards + '\n'
    + "     Bottom Mines =  " + subDetectBottMineRangeYards + '\n'
    + "     NOMBOs       =  " + subDetectNomboRangeYards + '\n';
  }
}
```

H. SUBMARINE LEAM

```
/*
 * SubmarineSimple.java
 * Created on September 10, 2003, 12:37 AM
 */

package submda;
import java.util.*;
import simkit.*;
import simkit.random.*;
import simkit.smdx.*;
import java.awt.geom.*;
import java.text.DecimalFormat;

// Uses Simple Turn One Direction Obstacle Avoidance (default is to the east)
// Main class should start this submarine on far west edge of minefield

public class SubmarineSimple extends SimEntityBase{

  // OUTLINE of code
  // Instance Variables (parameters and state variables)
  // Constructor
  // Instance Methods - Getters (for parameters and state variables)
  // Instance Methods - Setters (for parameters only)
  // Instance Methods - doEvents (doRun, doDetection...)
  // Instance Methods - Other (reset, toString...)

  // Instance Variables
  private double probDetect;        // used for coin flip for doDetection
                                            // not used in this version!
  DecimalFormat twoPlaces = new DecimalFormat("0.00");

  // Local Input Variables for Submarine:   Parameters (initialized in constructor)
  private Point2D.Double subStart;
  private Point2D.Double subFinish;
  private double newStart;          // used to set reentry point   // not used in this version
  private double subMaxSpeedKnots;
  private double subTransitSpeedKnots;
  private double subSpeedYdsHr; // units yards per hour
  private double meanTimeSubDetectContact;     // not used in this version
  private double subDetectMoorMineRangeYards;
  private double subDetectBottMineRangeYards;
  private double subDetectNomboRangeYards;    // (should be similar to range for bottom
            mines)
  private Sensor subSonarMoored;
  private Sensor subSonarBottom;
  private Sensor subSonarNombo;
  private Mover subMover;
  private PathMoverManager subMoverManager;
  private double msd;     //minimum standoff distance
  private double minefieldWidthYds;
  private double minefieldLengthYds;

  // Local Input Variables for Submarine:   State Variables (initialized in setter methods)
  private int numberDetections = 0;   // used to count firePropertyChange upon detections
  private boolean gridlock = false;  // only turns true when the submarine has tried to
          cross the right edge or bottom of minefield
  private double cX;   // current detected contact's X coordinate
  private double cY;   // current detected contact's Y coordinate

  // Constructor
  /** Creates a new instance of Sub */
  public SubmarineSimple(Point2D.Double start, Point2D.Double finish, double
          maxSpeed, double transitSpeed, double time, double detMoorRange, double
          detBottRange, double detNomboRange, double width, double length, double
          prob) {
    this.subStart = start;
    this.subFinish = finish;
    this.subMaxSpeedKnots = maxSpeed;
    this.subTransitSpeedKnots = transitSpeed;
    this.subSpeedYdsHr = subTransitSpeedKnots*2000.0;
    this.meanTimeSubDetectContact = time/60.0;              // not used in this version
    this.subDetectMoorMineRangeYards = detMoorRange;
    this.subDetectBottMineRangeYards = detBottRange;
    this.subDetectNomboRangeYards = detNomboRange;
    this.minefieldWidthYds = width*2000.0; // convert to yards
    this.minefieldLengthYds = length*2000.0; // convert to yards
    this.probDetect = prob;
    this.msd = subDetectMoorMineRangeYards;

    subMover = new UniformLinearMover("Sub", subStart,
          subMaxSpeedKnots*2000.0);    // mover is in units yds/hour not kts
    subMoverManager = new PathMoverManager(subMover, new ArrayList());
    subMoverManager.setStartOnReset(true);
    subMoverManager.addWayPoint(subFinish, subTransitSpeedKnots*2000.0);

    subSonarMoored = new ProbCookieCutterSensor(subDetectMoorMineRangeYards,
          subMover, probDetect);
    subSonarBottom = new ProbCookieCutterSensor(subDetectBottMineRangeYards,
          subMover, probDetect);
    subSonarNombo= new ProbCookieCutterSensor(subDetectNomboRangeYards,
          subMover, probDetect);
    // Submarine listens to all sensors (moored, bottom, and nombo sonars)
    subSonarMoored.addSimEventListener(this);
    subSonarBottom.addSimEventListener(this);
    subSonarNombo.addSimEventListener(this);
  }

  // Instance Method - Getter Methods (for states and parameters)
  public Point2D.Double getSubStart() {
    return subStart;
  }
  public Point2D.Double getSubFinish() {
    return subFinish;
  }
  public double getSubMaxSpeedKnots() {
    return subMaxSpeedKnots;
  }
  public double getSubTransitSpeedKnots() {
    return subTransitSpeedKnots;
  }
  public double getMeanTimeSubDetectContact() {
    return meanTimeSubDetectContact;
  }
  public double getSubDetectMoorMineRangeYards() {
    return subDetectMoorMineRangeYards;
  }
  public double getSubDetectBottMineRangeYards() {
    return subDetectBottMineRangeYards;
  }
  public double getSubDetectNomboRangeYards() {
    return subDetectNomboRangeYards;
  }
  public Sensor getSubSonarBottom() {
    return subSonarBottom;
  }
  public Sensor getSubSonarMoored() {
    return subSonarMoored;
  }
  public Sensor getSubSonarNombo() {
    return subSonarNombo;
  }
  public Mover getSubMover() {
    return subMover;
  }
  public PathMoverManager getSubMoverManager() {
    return subMoverManager;
  }
```

```
public int getNumberDetections() {
  return numberDetections;
}
public double getProbDetect() {
  return probDetect;
}
public boolean isGridlock() {
  return gridlock;
}

// Instance Method - Setter Methods:  for parameters only (not for state variables)
public void setSubStart(Point2D.Double point) {
  subStart = point;
}
public void setSubFinish(Point2D.Double point) {
  subFinish = point;
}
public void setSubMaxSpeedKnots(double speed) {
  subMaxSpeedKnots = speed;
}
public void  setSubTransitSpeedKnots(double speed) {
  subTransitSpeedKnots = speed;
}
public void setMeanTimeSubDetectContact(double time) {
  meanTimeSubDetectContact = time;
}
public void setSubDetectMoorMineRangeYards(double range) {
  subDetectMoorMineRangeYards = range;
}
public void setSubDetectBottMineRangeYards(double range) {
  subDetectBottMineRangeYards = range;
}
public void setSubDetectNomboRangeYards(double range) {
  subDetectNomboRangeYards = range;
}
public void setProbDetect(double prob) {
  ((ProbCookieCutterSensor)subSonarMoored).setProbDetect(prob);
  ((ProbCookieCutterSensor)subSonarBottom).setProbDetect(prob);
  ((ProbCookieCutterSensor)subSonarNombo).setProbDetect(prob);
}

// Instance Methods - doEvents (doRun, doDetection...)
public void doDetection(Moveable contact) {
  cX = contact.getLocation().getX();
  cY = contact.getLocation().getY();
  if ((cX+msd)<minefieldWidthYds){   // will not turn past right edge of minefield
```

```
    waitDelay("doTurnRight", 0.00001);
  }
  else {
    waitDelay("doGridlock", 0.0);
  }
}
public void doTurnRight() {
  subMoverManager.stop();
  subMoverManager.clearPath();
  firePropertyChange("numberDetections", numberDetections, numberDetections++);
  if (cY>=subMover.getLocation().getY()) {  // contact is north of sub
    subMoverManager.addWayPoint(new Point2D.Double(subMover.getLocation().
         getX(), cY-msd-1.0), subSpeedYdsHr);
    subMoverManager.addWayPoint(new Point2D.Double  (cX+msd+1.0,cY-msd-1.0),
         subSpeedYdsHr);
    subMoverManager.addWayPoint(new Point2D.Double
         (cX+msd+1.0,minefieldLengthYds), subSpeedYdsHr);
    subMoverManager.start();
  }
  else {      // contact is south of sub (detected while transiting East)
    subMoverManager.addWayPoint(new Point2D.Double(cX-msd-1.0,
         subMover.getLocation().getY()), subSpeedYdsHr);
    subMoverManager.addWayPoint(new Point2D.Double(cX-msd-1.0, cY-msd-1.0),
         subSpeedYdsHr);
    subMoverManager.addWayPoint(new Point2D.Double  (cX+msd+1.0,cY-msd-1.0),
         subSpeedYdsHr);
    subMoverManager.addWayPoint(new Point2D.Double
         (cX+msd+1.0,minefieldLengthYds), subSpeedYdsHr);
    subMoverManager.start();
  }
}
public void doGridlock() {         // effectively stops simulation since no more events
         scheduled after sub returns to bottom of minefield
  subMoverManager.stop();
  subMoverManager.clearPath();
  gridlock = true;
}

// Instance Methods - other than Getter, Setter, or "do"
// restore state variable to their initial values
public void reset() {
  super.reset();
  subMoverManager.clearPath();
  subMoverManager.addWayPoint(subFinish, subSpeedYdsHr);
  numberDetections = 0;
  gridlock = false;
  }
  public String paramString() {
    return "SUBMARINE ALWAYS TURNING ONE DIRECTION" + '\n'
    + "Sub's Start Position  = (" + twoPlaces.format(subStart.getX()) + ", " +
twoPlaces.format(subStart.getY()) + ") units of yards" +'\n'          // Try to show in units
of nm instead
    + "Sub's Finish Objective = (" + twoPlaces.format(subFinish.getX()) + ", " +
twoPlaces.format(subFinish.getY()) + ") units of yards" +'\n' + '\n'  // Try to show in units
of nm instead
    + "Sub's Max Speed (knots)     = " + subMaxSpeedKnots + '\n'
    + "Sub's Transit Speed (knots) = " + subTransitSpeedKnots + '\n'
    + "Sub's mean time to detect mine (minutes) = " + meanTimeSubDetectContact*60.0
+ '\n'
    + "Sub's probability of detection = " + probDetect + '\n'
    + "Sub's detection range (yards) for"  + '\n'
    + "     Moored Mines =  " + subDetectMoorMineRangeYards + '\n'
    + "     Bottom Mines =  " + subDetectBottMineRangeYards + '\n'
    + "     NOMBOs       =  " + subDetectNomboRangeYards + '\n';
  }
}
```

I. SIMULATE SUB MDA

```
package submda;

/*
 * SimulateSubMDA.java
 * Created on June 9, 2003, 10:59 AM
 * Terry Nawara
 */

import java.util.*;
import simkit.*;
import simkit.random.*;
import simkit.smdx.*;
import java.awt.geom.*;
import java.text.DecimalFormat;
import simkit.stat.*;
import simkit.util.*;
import java.io.*;

/* This program will instantiate a minefield and submarine.
 * Distances will be in units of yards.  Time units will be
 * in minutes. Therefore a sub moving at 5 kts (nm/hr)
 * will be moving at 10,000 yds/hr.
 *
 * One Sub with a Probability Cookie Cutter Sensor (A circular sensor
 * whose chance to detect is a given probability when the
 * target has entered its range) will move from bottom to
 * top of box.
 *
 * The box will have mines with Cookie Cutter Sensors (a circular
 * sensor who detects the target immediately after it enters within
 * range).
 */

public class SimulateSubMDA {

  // Main Method
  public static void main(String[ ] args) {

    // Variables, potential factors
    double minefieldWidthNm = 20.0;
    double minefieldLengthNm = 6.0;
    double bufferYds = 2000.0;     // number of yards the sub starts behind minefield as a
           buffer
    double subMaxSpeedKts = 30.0;
    double subTransitSpeedKts = 5.0;
    double prob = 0.0;

    // Statistics
    int replications = 0;
    double transitTime = 0.0;
    DecimalFormat twoPlaces = new DecimalFormat("0.00");
    DecimalFormat fourPlaces = new DecimalFormat("0.0000");
    SimpleStatsTally statProbSafeTransit = new SimpleStatsTally();
    SimpleStatsTally statTransitTime = new SimpleStatsTally();
    SimpleStatsTally statNumberDetections = new SimpleStatsTally();
    SimpleStatsTally statProbGridlocks = new SimpleStatsTally();
    double halfwidthProbSafeTransit = 0.0;
    double halfwidthTransitTime= 0.0;
    double halfwidthNumberDetections= 0.0;
    double halfwidthProbGridlocks= 0.0;

    // Simulation Controls
    boolean verboseModeOn = false;
    boolean reportSubParameters = false;
    boolean reportMinefieldParameters = false;
    boolean reportSingleTransits = false;

    // Instantiate Referees
    SensorTargetReferee refSub= new SensorTargetReferee();      // referee for moored &
           bottom mines detecting sub
    SensorTargetReferee refMoor = new SensorTargetReferee();    // referee for sub
           detecting moored mines
    SensorTargetReferee refBott= new SensorTargetReferee();     // referee for sub
           detecting bottom mines
    SensorTargetReferee refNombo = new SensorTargetReferee();   // referee for sub
           detecting nombos

    // Allow mines to be unregistered upon reset
    refSub.setClearOnReset(true);
    refMoor.setClearOnReset(true);
    refBott.setClearOnReset(true);
    refNombo.setClearOnReset(true);

    // Instantiate Minefield
    // Minefield inputs: width, length, moorDensity, bottDensity, nomboDensity,
           moorRange, bottRange
    PoissonMinefield myMinefield = new PoissonMinefield(minefieldWidthNm,
           minefieldLengthNm, 0.3, 0.0, 0.0, 999.0, 159.0);
```

```
// Instantiate Sub
Point2D.Double subStart = new Point2D.Double((minefieldWidthNm*2000.0)/2.0, -
       bufferYds);
// Finish should have the same X position as the start
Point2D.Double subFinish = new Point2D.Double((minefieldWidthNm*2000.0)/2.0,
       minefieldLengthNm*2000.0);
// Submarine inputs:  start, finish, max Speed, transit Speed, mean Detection time,
       detMoorRange, detBottRange, detNomboRange, width, length, prob detection
SubmarineSimple mySub = new SubmarineSimple(subStart, subFinish, 30.0,
       subTransitSpeedKts, 2.0, 1000.0, 160.0, 160.0, minefieldWidthNm,
       minefieldLengthNm, prob);

mySub.getSubMoverManager().setStartOnReset(true);

// Assign Sensor and Mover to Mediators
SensorTargetMediatorFactory.getInstance().addMediatorFor(
Mine.class, simkit.smdx.UniformLinearMover.class,
simkit.smdx.CookieCutterMediator.class);

SensorTargetMediatorFactory.getInstance().addMediatorFor(
ProbCookieCutterSensor.class, simkit.smdx.UniformLinearMover.class,
ProbCookieCutterMediator.class);

// loop ensures enough runs to get desired confidence interval for Prob safe  transit
do {
  replications++;
  // Clear counters
  transitTime = 0.0;
  Schedule.reset();

  // Register sensors with referee for sub // this sets up 3 referees (one per mine type)
  refMoor.register(mySub.getSubSonarMoored());
  refBott.register(mySub.getSubSonarBottom());
  refNombo.register(mySub.getSubSonarNombo());
  // Register submarine as a target for referee where mines are sensors
  refSub.register(mySub.getSubMover());

  // Register movers & sensors with referee for mines
  for (int i=0; i< myMinefield.getTotalMoorMines(); i++) {
    refMoor.register(myMinefield.getMoorMine(i).getMover());
    refSub.register(myMinefield.getMoorMine(i));
  }
  for (int i=0; i< myMinefield.getTotalBottMines(); i++) {
    refBott.register(myMinefield.getBottMine(i).getMover());
    refSub.register(myMinefield.getBottMine(i));
  }
```

```
    for (int i=0; i< myMinefield.getTotalNombos(); i++) {
      refNombo.register(myMinefield.getNombo(i).getMover());
    }

    // Invoke Simulation
    Schedule.setVerbose(verboseModeOn);
    Schedule.startSimulation();
    transitTime = Schedule.getSimTime();

    // Output
    if (reportSingleTransits) {
      System.out.println('\n' + "RESULTS OF " + replications + "th MINEFIELD
         TRANSIT" + '\n');
      System.out.println("The number of moored mines actuated was " +
         myMinefield.getTotalNumberMoorMinesActuated());
      System.out.println("The number of bottom mines actuated was " +
         myMinefield.getTotalNumberBottMinesActuated());
      System.out.println("The run time was " + fourPlaces.format(transitTime) + "
         hours");
      System.out.println("That run time is equivalent to " +fourPlaces.format(
         transitTime *subTransitSpeedKts) + " nm, given transit speed of " +
         subTransitSpeedKts + " knots");
      System.out.println("The number of MILCO's detected by submarine was " +
         mySub.getNumberDetections());
    }

    // Collect Statistics
    // Determine if transit resulted in gridlock, safe (no mine actuation), or kill
    if (mySub.isGridlock()) {
      //System.out.println("The submarine is gridlocked");
      statProbGridlocks.newObservation(1.0);
      statProbSafeTransit.newObservation(0.0);
    }
    else if (myMinefield.getTotalNumberMoorMinesActuated()
        +myMinefield.getTotalNumberBottMinesActuated() == 0){
      statProbSafeTransit.newObservation(1.0);
      statTransitTime.newObservation(transitTime);
      statProbGridlocks.newObservation(0.0);
    }
    else {
      statProbGridlocks.newObservation(0.0);
      statProbSafeTransit.newObservation(0.0);
    }

statNumberDetections.newObservation(mySub.getNumberDetections());
```

```
     halfwidthProbSafeTransit = (statProbSafeTransit.getStandardDeviation()
            /Math.sqrt(statProbSafeTransit.getCount())) * StudentT.getQuantile(0.975,
            statProbSafeTransit.getCount() -1);
     halfwidthTransitTime = statTransitTime.getStandardDeviation()/
            Math.sqrt(statTransitTime.getCount())) * StudentT.getQuantile(0.975,
            statTransitTime.getCount() -1);
      halfwidthNumberDetections = (statNumberDetections.getStandardDeviation()
            /Math.sqrt(statNumberDetections.getCount())) * StudentT.getQuantile(0.975,
            statNumberDetections.getCount() -1);
       halfwidthProbGridlocks = (statProbGridlocks.getStandardDeviation()/
            Math.sqrt(statProbGridlocks.getCount())) * StudentT.getQuantile(0.975,
            statProbGridlocks.getCount() -1);
    }while (halfwidthProbGridlocks>0.1||replications<30);
    // Output for batch runs
    if (reportSubParameters) {
      System.out.println(mySub.paramString());
    }
    if (reportMinefieldParameters) {
      System.out.println(myMinefield.paramString());
    }
    System.out.println('\n' + "RESULTS OF MINEFIELD TRANSIT AFTER " +
            replications + " RUNS" + '\n');
    System.out.println("The average probability of gridlock = " +
            fourPlaces.format(statProbGridlocks.getMean()));
    System.out.println("Sub's Pd was = " + prob);
    System.out.println("Avg gridlock = " +
            fourPlaces.format(statProbGridlocks.getMean()));
    System.out.println("The average prob of kill (Pk) = 1 - Pst - Pgridlock = " +
            fourPlaces.format(1.0-statProbGridlocks.getMean()-
            statProbSafeTransit.getMean()));
    System.out.println("The average probability of safe transit (Pst) = " +
            fourPlaces.format(statProbSafeTransit.getMean()));
    System.out.println("Halfwidth for Pst =                  " +
            fourPlaces.format(halfwidthProbSafeTransit) + '\n');
    System.out.println("The average transit time for safe transits was = " +
            fourPlaces.format(statTransitTime.getMean()) + " hours");
    System.out.println("Remember, the sub starts back " + bufferYds + " yards so it
            doesn't start on top of a mine.");
    System.out.println("Halfwidth for transit time for safe transit is = " +
            fourPlaces.format(halfwidthTransitTime));
    System.out.println("The average number of detections by the submarine was = " +
            fourPlaces.format(statNumberDetections.getMean()));
    System.out.println("Halfwidth for number of detections by the submarine   = " +
            fourPlaces.format(halfwidthNumberDetections));
  }
}
```

LIST OF REFERENCES

"AN/BLQ-11 Long Term Mine Reconnaissance System (LMRS)," NWP 3-15.5 TP, Draft, November 28, 2001.

"An Introduction to Percolation and Many-body Physics by Computer." http://fafnir. Phyast.pitt.edu/myjava/perc/pretest.html. [17 September 2003].

Boerman, Douglas A. "Finding an Optimal Path Through a Mapped Minefield," M.S. Thesis, Naval Postgraduate School, Monterey, California, March 1994.

Buss, Arnold. "Basic Event Graph Modeling." Simulation News Europe. Issue 31, April 2001.

Buss, Arnold. "Component-Based Simulation Modeling," Proceedings of the 2000 Winter Simulation Conference, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds., 2000.

Buss, Arnold. "Discrete Event Modeling with Simkit." Simulation News Europe. Issue 32/33. November 2001.

Buss, Arnold. "Modeling with Event Graphs," Proceedings of the 1996 Winter Simulation Conference, J. M. Games, D. J. Morrice, D. T. Brunner, and J. J. Swain, eds., 1996.

Buss, Arnold. "Simkit." http://diana.gl.nps.navy.mil/Simkit/ June 2003.

Crimmins, Jim. "Mine Warfare and Submarines," U.S. Naval Institute Proceedings, October 1994: 80-81.

Devore, Jay L. Probability and Statistics for Engineering and the Sciences. 5th Ed., Duxbury, 2000.

Frey, Tim and Ethan Decker. “Percolation Theory.” Ecological Complexity Seminar: Fall 1996, University of New Mexico, <http://sevilleta.unm.edu/~ehdecker/complexity/96fall/percol.html> [15 September 2003].

Gaver, Donald P., Patricia A. Jacobs, and Steven E. Pilnick, “Obstacle Avoidance.” Unpublished Research Proposal, Naval Postgraduate School, March 2003.

Gilman, George, J. Marc Eadie, and Hai Tran. “Autonomous Littoral Warfare Systems Evaluator (ALWSE) User Manual” ALWSE-MC-01-UG-001, June 2003.

Grossenbacher, John J. “Remarks at 2002 NDIA Clambake,” The Submarine Review, (January 2003): 6-14.

Grossenbacher, John J. “SUBLANT: Anytime, Anywhere,” Undersea Warfare, Vol. 3, NO. 1, Fall 2000.

Hyland, John C. “A Reflexive Mine Avoidance Approach for Autonomous Underwater Vehicles.” PhD Dissertation, University of Florida, 1993.

Hyland, John C. and Fred J. Taylor, “Mine Avoidance Techniques for Underwater Vehicles,” IEEE Journal of Oceanic Engineering, Vol. 18, NO. 3, July 1993, 340-350.

Kim, Chihoon. “The Effect of Sensor Performance on Safe Minefield Transit,” M.S. Thesis, Naval Postgraduate School, December 2002.

Law, A. and D. Kelton.. Simulation Modeling and Analysis, Third Edition, McGraw-Hill, Boston. MA.. 2000.

Meester, Ronald and Rahul Roy. Continuum Percolation. Cambridge University Press, Cambridge. 1996.

Natter, Robert J. “Access in Not Assured,” U.S. Naval Institute Proceedings, (January 2003): 39-41.

Naval Operations Analysis. 2rd Ed. Annapolis: Naval Institute Press, 1977.

Naval Studies Board. “Committee for Mine Warfare Assessment.” 2001.

Pollitt, George, Ian Craig, Joe Gezelter, Lance Hereford, “COMID 2002 Warfighter Payoff Analysis: Fusion Applications for Transit Deep Water SLOC,” 29 August 2002.

Ross, Sheldon M., Introduction to Probability Models, 7th Ed., Academic Press, 2000.

Sanchez, Susan. “ABC’s of Output Analysis.” Proceedings of the 2001 Winter Simulation Conference, B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, eds., 2000: 30-38.

San Jose, Angel. “Analysis, Design, Implementation and Evaluation of Graphical Design Tool to Develop Discrete Event Simulation Models Using Event Graphs and Simkit.” M.S. Thesis, Naval Postgraduate School, September 2001.

Schaffer, Matt. “An Analysis of Minefield Avoidance and Penetration.” Center for Naval Analyses, CRM 90-18, June 1990.

Schaffer, Matt. “Follow-On Analysis of Minefield Avoidance and Penetration.” Center for Naval Analyses, CRM 91-40, 1991.

Schinazi, Rinaldo B. Classical and Spatial Stochastic Processes. Boston: Birkhauser, 1999.

Taylor, Howard M. and Samuel Karlin. An Introduction to Stochastic Modeling. 3rd Ed. San Diego: Academic Press, 1998.

Tychonievich, Lou, Thomas C. Smith, and Robert Evans, “Influence Networks: A Reactive Planning Architecture.” Proceedings Seventh IEEE Conference on Artificial Intelligence for Applications, 1991: 354-360.

U.S. Navy Department. AN/BLQ-11 Long Term Mine Reconnaissance System (LMRS). NWP 3-15.5 (Draft): G1-G10.

Johns Hopkins Applied Physics Laboratory. Mission Requirements and Requirements Analysis for the Advanced Vehicle Concept. JWR-97-017. Laurel, MD, 1992.

Johns Hopkins Applied Physics Laboratory. Need for and Utility of a Submarine Offboard Mine Search System (SOMSS). Laurel, MD. 30 August 1993.

Wagner, Daniel H., W. Charles Mylander, and Thomas J. Sanders, ed. Naval Operations Analysis. 3rd Ed. Annapolis: Naval Institute Press, 1999.

Washburn, Alan R., Search and Detection, 3rd Ed., Institute for Operations Research and the Management Sciences, 1996.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Steven E. Pilnick
   Naval Postgraduate School
   Monterey, California

4. Arnold H. Buss
   Naval Postgraduate School
   Monterey, California

5. Pat Madden
   Johns Hopkins University Applied Physics Laboratory
   Laurel, Maryland

6. VADM Roger Bacon
   Chair of Undersea Warfare
   Naval Postgraduate School
   Monterey, California

7. RADM John D. Pearson
   Chair of Mine Warfare
   Naval Postgraduate School
   Monterey, California

8. CAPT Jeff Kline
   Naval Postgraduate School
   Monterey, California

9. Alan R. Washburn
   Naval Postgraduate School
   Monterey, California

10. Patricia A. Jacobs
    Naval Postgraduate School
    Monterey, California

11. Donald P. Gaver
Naval Postgraduate School
Monterey, California

12. W. Matthew Carlyle
Naval Postgraduate School
Monterey, California

13. LCDR Kelly J. Cormican
Naval Postgraduate School
Monterey, California

14. Jerry Rosenberger
Institute for Defense Analyses
Alexandria, Virginia

15. LCDR Barry Ryan
Submarine Development Squadron Twelve
N77 - Submarine Survivability/MCM
Groton, Conneticut

16. Steve Boothe
Lead Analyst
Undersea Warfare Division
COMOPTEVFOR Code 40B2

17. Brenna Williams
Head, MIW Tactics and Analysis Branch, A83
Coastal Systems Station
Dahlgren Division
Naval Surface Warfare Center
Naval Sea Systems Command

18. H. Sam Richardson III
Senior Sonar Systems Engineer
Coastal Systems Station
Dahlgren Division
Naval Surface Warfare Center
Naval Sea Systems Command